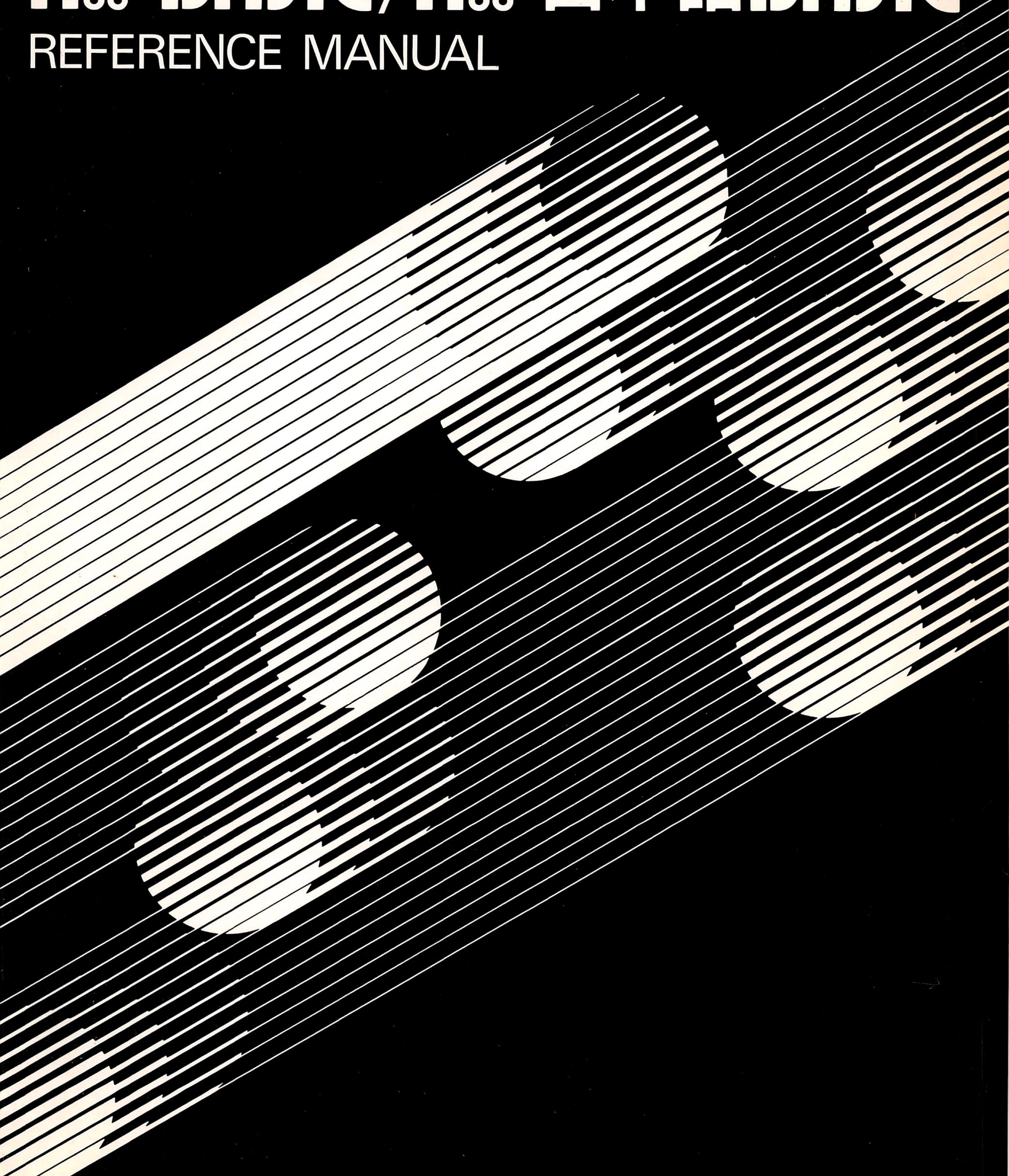
NEC Personal Computer

PC-8801FH/MH

N88-BASIC/N88-日本語BASIC

REFERENCE MANUAL

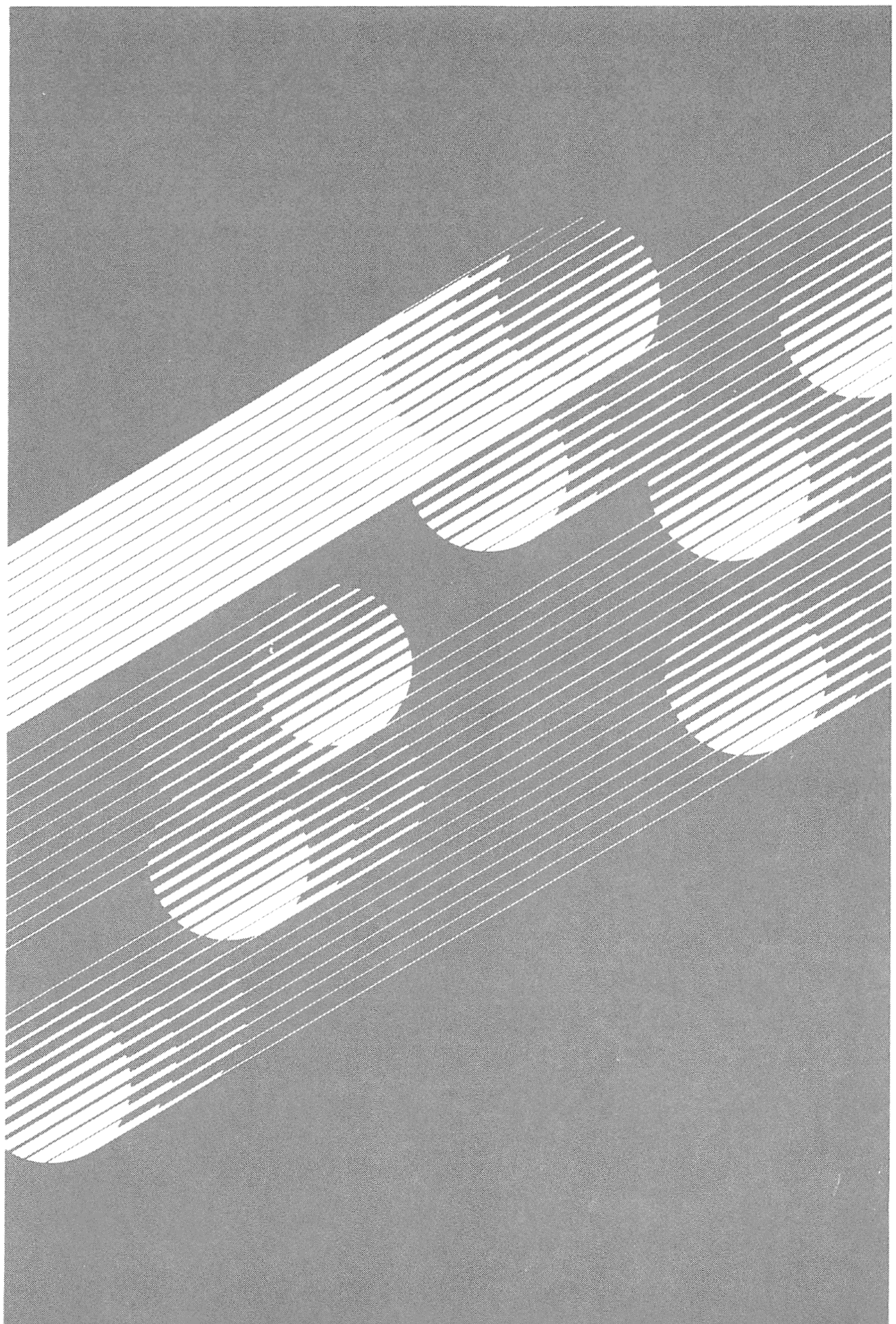


PC-8801 FH/MH-RM

PC-8801FH/MH

N_{88}-BASIC/N_{88}-日本語BASIC

REFERENCE MANUAL



まえがき

パーソナルコンピュータPC-8801FH/PC-8801MH(以下本機と呼びます)は，プログラミング言語としてN₈₈-BASICとN₈₈-日本語BASICの2つのBASICを搭載しています．

本書は，ある程度プログラミングの知識のある方を対象にその機能を解説しています．

このマニュアルを読む前に，本機ユーザーズガイドを読んで，基本的な操作方法を習得することをお勧めします．

本書を熟読され，本機の機能が十分に活かされたBASICのソフトウェアが数多く作り出されることを期待いたします．

このマニュアルの構成

本書はN_{88}-BASIC/N_{88}-日本語BASICの各機能，命令などをわかりやすく解説するために，次のような構成になっています．

第1章　N_{88}-BASIC/N_{88}-日本語BASICの文法

N_{88}-BASIC/N_{88}-日本語BASICの文法をコマンド・ステートメントなどの説明に先だって解説しています．その内容は，BASIC言語の定数，変数の意味，演算の決まりなどBASIC言語を使う上での最低限の知識です．これらは2章，3章の文法の説明を理解する上でどうしても必要な事項です．必ず読むようにしてください．

第2章　基本命令

N_{88}-BASIC V1, V2およびN_{88}-日本語BASICで使うことができるコマンド・ステートメント・関数について説明します．

第3章　拡張命令

N_{88}-BASIC/N_{88}-日本語BASICでは，さらに拡張命令を追加することができます．

この章では，拡張命令の概要，追加手順，メモリマップそして追加されたステートメントについて説明します．

資　料

N_{88}-BASIC/N_{88}-日本語BASICの予約語やエラーメッセージ，コントロールコード表などプログラミングに必要な資料で構成されています．

このマニュアルで使われている記号

このマニュアルでは解説をわかりやすくするために，次の記号を使っています．

[↵]　リターンキーの入力を表します．

[ ]　スペースを表します．

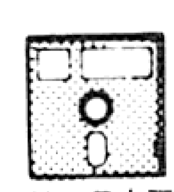
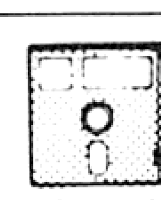
このマニュアルで使われている用語

本機には強力な２つのBASIC——N_{88}-BASICとN_{88}-日本語BASIC——が装備されています．

N_{88}-BASICはいくつかのバージョンを持ち，またディスク装置を使う場合，使わない場合などでも少しずつ機能が異なります．

ここでは，BASICのバージョンやモードを中心として，このマニュアルで使われている用語をまとめておきます．

1. BASICの種類やモード名

用　　語	意　　味	参照マニュアル，章，シンボル
N_{88}-BASIC V1	従来のN_{88}-BASIC（PC-8801，PC-8801$_{MK}$II 用のN_{88}-BASIC）と互換性があるモード． 本機のシステムモードスイッチをV1SまたはV1Hにセットしてスタートさせる． N_{88}-BASIC システムディスクを使ってスタートさせるV1 DISK とディスクを使わない V1 ROM の２つのモードの総称．	ユーザーズガイド 第４章 プログラマーズガイド 第１章
N_{88}-BASIC V2	PC-8801$_{MK}$IISR，FR, MR, TR 用の，N_{88}-BASICと互換性のあるモード． 本機のシステムモードスイッチをV2にセットしてスタートさせる． N_{88}-BASICシステムディスクを使ってスタートさせるV2 DISKと，ディスクを使わないV2 ROMの２つのモードの総称．	ユーザーズガイド 第４章 プログラマーズガイド 第１章
N_{88}-日本語BASIC	N_{88}-BASIC V2に，日本語処理機能を付加して，PC-8801$_{MK}$IIFR，MR，PC-8801FH, MH 用に開発されたBASIC． 本機のシステムモードスイッチをV2にセットし，N_{88}-日本語BASICシステムディスクを使ってスタートさせる． 5.25インチディスクドライブにシステムディスクをセットした状態で動作する．	ユーザーズガイド 第４章 プログラマーズガイド 第１章 
N_{88}-BASIC DISK version	V1 DISKとV2 DISKの総称．	ユーザーズガイド 第４章 プログラマーズガイド 第１章
N_{88}-BASIC ROM version	V1 ROMとV2 ROMの総称．	
BASIC	パソコンで一般的に広く使われているBASICすべての総称． 狭義には本機のN_{88}-BASIC V1，V2，N_{88}-日本語BASICを意味する．	
拡張命令	N_{88}-BASIC V2およびN_{88}-日本語BASICで，NEW CMDコマンドを実行することにより，使用可能になるサウンド機能やアナログパレット機能をサポートする命令群．	BASICリファレンスマニュアル 第３章

2. N88-日本語BASICで使われる用語

用　　語	意　　味	参照マニュアル，章
日本語モード	SCREEN文の第1パラメータを3，または4に指定したときのモード．全角文字の画面表示やキー入力が可能．N88-日本語BASICのスタート時は，標準ディスプレイ使用時はSCREEN 4，専用ディスプレイ使用時はSCREEN 4に設定されている．	プログラマーズガイド 第5章，第6章
グラフィックシンボルモード	日本語モードに対するモードで，SCREEN文の第1パラメータを，0，1，2に指定したときのモード．全角文字の画面表示，キー入力はできない．全角文字に相当するコードを表示させると，グラフィックシンボル(BASICリファレンスマニュアル　資料5)を含む半角文字が表示される．	プログラマーズガイド 第8章
全角文字	日本語を表すための文字(BASICリファレンスマニュアル　資料8　日本語コードにあげられている文字).BASICはシフトJISコードによって全角文字を扱い，MID$などの関数は，全角文字1文字を半角文字2文字と等価に扱う．	プログラマーズガイド 第5章，第6章 BASICリファレンスマニュアル　資料7
半角文字	英数字，カタカナ，コントロールコードを含む文字(資料4　キャラクタコードにあげられている文字)．N88-BASICなどの日本語処理機能を持たないBASICでは半角文字しか使用できない．	プログラマーズガイド 第5章，第6章 BASICリファレンスマニュアル　資料7
全角文字入力モード 半角文字入力モード	日本語モードでは，全角文字と半角文字とを混在して使用することができる．キー入力時には [全角] (全角キー)によって，全角文字を入力するためのモード(全角文字入力モード)と，半角文字を入力するモード(半角文字入力モード)とを切り替えながら行う．	プログラマーズガイド 第5章

N88-BASIC/N88-日本語BASIC
REFERENCE MANUAL
目　　次

第1章　N88-BASIC/N88-日本語BASICの文法

第2章　基本命令

第3章　拡張命令

資　料

機能別索引

1．一般的な命令

一般コマンド

一般ステートメント

2. 入出力に関する命令・関数

入出力コマンド

入出力ステートメント

フロッピーディスク

カセットテープ

画面・キーボード・プリンタ

入出力関数

フロッピーディスク

キーボード・プリンタ

RS-232C通信ポート

3．画面制御に関する命令・関数

画面制御ステートメント

画面制御関数

画面制御予約変数

4．グラフィックスに関する命令・関数

グラフィックステートメント

グラフィック関数

5. 算術演算に関する関数

数値関数

6. 文字列操作に関する命令・関数

文字ステートメント

文字関数

7. 特殊な命令や関数

特殊コマンド

特殊ステートメント

特殊関数

予約変数

8．日本語処理に関する命令・関数

日本語ステートメント

日本語関数

9．拡張命令

ステートメント

第1章

N_{88}-BASIC／N_{88}-日本語BASICの文法

この章ではN_{88}-BASIC/N_{88}-日本語BASIC言語の基本的な規則について解説します．内容は，BASIC言語の定数，変数の意味，演算の決まりなどBASIC言語を使う上で最低限必要とされる知識です．またこれらは2章，3章の命令を理解するのに必要な事項です．

1.1 動作モード

BASICを起動すると(起動方法については**ユーザーズガイド**第4章「BASICを使おう」を参照してください)，画面に"Ok"という文字が表示されて，BASICはコマンドレベルになります．この状態のときは，あらゆるBASICのコマンドをキーボードから入力できます．

1.1.1 ダイレクトモード

行番号(1～65529まで)をつけずにBASICの文法に沿った文を入力した場合，その文はキャリッジリターンを入力後，すぐに実行されます．これをダイレクトモードにおける実行といいます．

1.1.2 プログラムモード

行番号(1～65529)をつけて文を入力した場合，それはメモリの中にプログラムとして行番号とともに格納されます．そしていったん格納されたプログラムは，RUNコマンドおよびGOTO文，GOSUB文によって実行させることができます．これをプログラムモードによる実行と呼びます．

1.2 文

文とは，BASICが行う手続きを記述している最小単位です．

文には，BASICが実行する式，ステートメント，コマンド，関数などを書くことができます．また文は，コロン(：)を用いて他の文とつなぐことができます．これは複文(マルチステートメント)と呼ばれ，複文は，1行(行番号を含めて)255文字以内の長さまで許されています．マルチステートメントについては1.5　特殊記号の使い方を参照してください．

1.3 行番号

BASICの各プログラム行は，行番号で始まらなければなりません．行番号には，1～65529までの整数を用います．行番号はプログラム行をメモリに格納する順序を示し，実行も行番号の若い方から行われます．また分岐や編集の目印としても使用されます．

行番号の代わりにピリオド(.)を使うことができる場合があります．ピリオドはエラー発生，編集時において現在の行を表す行番号の代わりとして，LIST, AUTO, DELETE, EDITなどで使用できます．

例)
```
10 PRINT "abcd"
20 CMD
30 PRINT "efgh"
RUN
abcd
Feature not available in 20
Ok
LIST□.注意
20 CMD
Ok
```
注意：□はスペースを表わします(本文P1-6　スペース参照)．

1.4 ラベル

BASICでは，飛び先を行番号で指定する代わりにラベルで指定することができます．次のプログラムを見てください．

例)
```
10 INPUT A
20 IF A<0 THEN 80
30 IF A>0 THEN 60
40 PRINT "zero"
50 GOTO 90
60 PRINT "plus"
70 GOTO 90
80 PRINT "minus"
90 END
```

これは，10行で入力された値が正か負かゼロかを調べるプログラムです．ここで処理の流れを変える命令が20，30，50，70行に使われていて，その飛び先の指定はすべて行番号になっています．しかし，この行番号はあくまで数字の並びであり，我々にはなじみにくく，プログラムの作成，デバッグ時においてかなり混乱を招きます．

N88-BASIC/N88-日本語BASICでは，飛び先の指定にラベル名を使うことができます．ラベル名を使って先ほどのプログラムを書き換えてみましょう．

例）
```
10 INPUT A
20 IF A<0 THEN *MINUS
30 IF A>0 THEN *PLUS
40 PRINT " zero "
50 GOTO *EXIT
60 *PLUS : PRINT " plus "
70 GOTO *EXIT
80 *MINUS : PRINT " minus "
90 *EXIT : END
```

このようにラベル名とは，ユーザが分岐先の目印として独自につけることのできる名前なのです．

このラベル名の使用においては，次の注意を守らなければなりません．

1. ラベル名の頭には必ずアスタリスク(*)をつけなければなりません．
2. 先頭のアスタリスクを除いて，ラベル名は必ず英文字で始まらなければなりません．
3. ラベル名に使用できる文字は，先頭のアスタリスクを除いて英文字と数字とピリオド(．)であり，大文字と小文字の区別はありません．
4. BASICの予約語(資料１　N88-BASIC/N88-日本語BASICの予約語参照)をラベルとして使用することはできません．ただし，中に含む場合はかまいません．
5. ラベル名の長さは，プログラムの１行の許可範囲(１行255文字)によってのみ制限を受けます．
6. 参照されるラベル名(呼ばれる方のラベル名)は，必ず行の最初になければなりません．
7. １行中でラベル名の後に続けて命令を記述し，マルチステートメントとするときは，コロン(：)またはスペースによって区切ります．

以上のような制限を無視すると，プログラムのロードおよび実行直後に"Syntax error"が表示されます．(行番号は表示されません．) この場合，LISTコマンドやEDITコマンドは実行できますから，ラベル名を訂正してください．

この他参照されるラベル名に同じものがあった場合，２重定義されているという意味の"Duplicate label"エラーが生じます．これらのエラーの検出は他のエラーメッセージの場合と異なり，"RUN"したときプログラムの実行に先だって行われますから，ラベル名のエラーがあるとプログラムを全く実行することができません．この場合は，LISTコマンドまたはEDITコマンドを使用してプログラムを表示させ，間違えたラベル名をスクリーンエディタ(**プログラマーズガイド**　第４章「スクリーン・エディタ」参照)を使ってなおしてください．

また，この場合のエラーメッセージは，エラー箇所を示す〈行番号〉は伴いません．

ラベル名は，プログラム中で分岐の目印として使用する他，コマンドレベルで"LIST"，"DELETE"など〈行番号〉を対象とするパラメータにはすべて使用することが可能です．

例)
```
10 INPUT A
20 IF A=0 THEN *EXIT ELSE *CALC
30 *CALC
40 MLT=A*A:PRINT MLT
50 GOTO 10
60 *EXIT
70 END
```

LISTコマンドを使用して*CALCから*EXITまでの行を表示します．

```
LIST *CALC-*EXIT
30 *CALC
40 MLT=A*A : PRINT MLT
50 GOTO 10
60 *EXIT
```

1.5 特殊記号の使い方

BASICでは，演算子(+，-，*，/)などの他にも特別な意味を持つ記号があります．

1. ダブルクォーテーション(")

ダブルクォーテーションで囲まれた内容を文字列として扱います．文字をダブルクォーテーションで囲んで表示させた場合と，囲まずに表示させた場合の違いは次のようになります．

例)
```
10 ABC=123
20 PRINT "ABC"    ←――文字列として扱う
30 PRINT ABC      ←――変数として扱う
40 END
RUN
ABC
 123
Ok
```

2. ピリオド(.)

ピリオドはエラーが発生した行，または新しく入力した行の行番号の値を持ちます．LIST, DELETE, AUTO, EDITなどで使えます．

例)　LIST□.

3. ハイフン(−)

LIST, DELETE命令など行の範囲を指定するとき，何行から何行までという場合に使います．

例)　DELETE 100−200

4. コロン(：)

マルチステートメントの区切りとして使います．マルチステートメントとは，1行(255文字以内)に複数の命令を書いてあるものです．

例)
```
10 A=B+C
20 PRINT A
```
＝ 10 A=B+C:PRINT A

5. コンマ(,)

PRINT, INPUTなどパラメータが並ぶ場合その区切りとして多用されます．

例)
```
INPUT A,B,C
CONSOLE,,,1
```

6. セミコロン(；)

PRINT文などの区切りとして使います．この場合，前の表示の後に続けて表示することができます．

例)　PRINT "A= ";A

7. アポストロフィ(’)

REM文(リマーク)の代用として使えます．

例)
```
REM TEST
'SAMPLE
```

8. クエスチョンマーク(？)

PRINT文の代用として使えます．

例)
```
?2*2
 4
```

9. アスタリスク(＊)

ラベル名の先頭につけます．

例）

```
10 INPUT A
20 GOSUB ＊SQUARE
30 GOTO 10
40 ＊SQUARE
50 PRINT A ^ 2
60 RETURN
```

10. スペース

BASICは原則としてスペースを無視します．ただしコマンド，ステートメントの後には，必ずスペースを入れなければなりません．（本書ではスペースを□で示してあります.）

例）

```
PRINT□A
LIST□100
```

1.6 定　数

1.6.1 定数の種類

N_{88}-BASIC/N_{88}-日本語BASICの定数には以下のものがあります．

- 定数
  - 文字定数（日本語文字も含む）
  - 数値定数
    - 整数型
      - 10進形式
      - 8進形式
      - 16進形式
    - 実数型
      - 単精度型
      - 倍精度型

1.6.2 文字定数

文字定数とは，255文字以下のダブルクォーテーション（"）で囲まれた英数字，カナ文字，記号および日本語文字などの列のことです．なお，（"）を文字列の中に記述する場合は，CHR$関数を用いなければなりません．

例）

```
PRINT " Good Morning "
Good Morning
PRINT " 100＋20 "
100＋20
```

1.6.3　数値定数

数値定数は，整数型，実数型に分けられ，それらは正あるいは負の数，または0です．

負の数の前には必ず符号をつけなければなりませんが，正の数の前の符号は省略することができます．

1.6.4　整数型

(1)　10進形式

－32768から＋32767までのすべての整数．または，数の後に％(1.7.2　変数名と型宣言文字参照)をつけたもの．

小数点をつけることはできません．

例)　32767

－123

32767％←─整数を表す．

(2)　8進形式

頭に&Oまたは＆を伴った，0から7までの数字の並び．

＆0～＆177777の範囲

例)　&12345 (5349(10))

&O7777(4095(10))

注意：(10)は10進数を表します．

(3)　16進形式

頭に＆Hを伴った，0からFまでの並び．&H0～&HFFFFの範囲．

例)　&H100(256(10))

&H7FFF(32767(10))

注意： 8進形式または16進形式で入力された数値は，PRINT, LPRINTでは10進形式で出力されます．

例）　10進数，8進数，16進数の関係

10進数	8進数	16進数
0	&0	&H0
1	&1	&H1
2	&2	&H2
3	&3	&H3
4	&4	&H4
5	&5	&H5
6	&6	&H6
7	&7	&H7
8	&10	&H8
9	&11	&H9
10	&12	&HA
11	&13	&HB
12	&14	&HC
13	&15	&HD
14	&16	&HE
15	&17	&HF
16	&20	&H10
17	&21	&H11
18	&22	&H12
19	&23	&H13
:	:	:
60	&74	&H3C
61	&75	&H3D
62	&76	&H3E
63	&77	&H3F
64	&100	&H40
65	&101	&H41
66	&102	&H42
67	&103	&H43
68	&104	&H44
69	&105	&H45
:	:	:
250	&372	&HFA
251	&373	&HFB
252	&374	&HFC
253	&375	&HFD
254	&376	&HFE
255	&377	&HFF
256	&400	&H100
257	&401	&H101
258	&402	&H102
259	&403	&H103
:	:	:

1.6.5　実数型

実数型は，単精度型と倍精度型に分けられます．

1.　単精度型

有効桁7桁の精度で格納されます．出力のときは7桁目が四捨五入され，6桁以下で表示されます．扱える数値は－1.70141E＋38～1.70141E＋38です．

(1)　7桁以下の実数

(2)　Eを使った指数形式

(3)　最後に！(単精度実数型を表す型宣言文字)を伴った数

例)　1.23

－7.09E－06＝-7.09×10^{-6}＝－0.00000709

3525.68

3.14!

2.　倍精度型

有効桁16桁の精度で格納され，16桁以下で表示されます．扱える数値は単精度型と同様です．

(1)　8桁以上の数

(2)　Dを使った指数形式

(3)　最後に#(倍精度実数型を表す型宣言文字)を伴った数

例)　1234567890

0.3141592653D＋01＝0.3141592653×10^{1}＝3.141592653

56789.0#

8657036.1543976

1.7 変　数

1.7.1 変数とは？

変数とは，数値や文字列を記憶させておく場所で，英数字からなる名前(たとえば，COUNT, XPOSなど)がついています．この名前を変数名と呼びます．

この変数を使うことによって，ある値を記憶させておいたり，取り出したり，あるいは参照したりすることができます．

例)　PI=3.14159　←──記憶させる

　　PRINT PI*2　←──取り出す

定数と同じように変数にも数値型(数値変数)と文字型(文字変数)があります．数値変数は，数値を記憶させておくもので，文字列を入れることはできません．逆に，文字変数に数値を入れることもできません．数値変数は，値が割り当てられる前に参照されたら0が代入され，文字変数はヌルストリング(空の文字列)が代入されます．

次の例は悪い例です("Type mismatch"エラーとなります．)

例)　PI= " BAD EXAMPLE "

　　NAME$=3.14159

1.7.2 変数名と型宣言文字

変数名は英字で始まる最大40文字の英数字とピリオド(．)で表され，そのすべてを区別します．

例1)　ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890123A

　　ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890123B

　　この2つは違った変数名とみなされます．

変数名は予約語(資料1　N_{88}-BASIC/N_{88}-日本語BASICの予約語参照)であってはなりませんが，予約語を含んだものはかまいません．ただし，FNで始まる変数名は許されません．また英文字において大文字，小文字の区別はありません．

例2)　RUN　←── 予約語のため変数名とはなりません．

　　RUNI ｝←── この場合は予約語を含んだ形となるので
　　XRUN ｝　　変数名として使えます．

同じ変数名でも型が違えば区別されます．変数の型は型宣言文字によって決まり，型宣言文字は変数名の最後につけて，その変数の型を表します．型宣言文字を省略すると，"！"がついているとみなされます(単精度実数型変数となる)．

型宣言文字
- % 整数型
- ! 単精度実数型
- # 倍精度実数型
- $ 文字型

例3) A
A#
A%
A$ ― これらは区別されますが，AとA！は同じで単精度実数型を意味します．

変数の型を宣言するのにもう一つ便利な方法があります．それはDEFINT，DEFSNG，DEFDBL，DEFSTRの型宣言文をプログラム中で用いる方法です．これについては2章で詳しく説明します．

1.7.3 配列変数

配列とは，同じ性格のデータが複数個集まったものであり，その配列を記憶させておく場所を配列変数といいます．

配列変数はDIM文（2章参照）で宣言し，要素（データ）は添字によって順序付けられます．次元は添字の最大値の個数によって決まり，1次元から多次元まで指定できます．

例）

1次元配列

DIM A(5)

A(0)
A(1)
A(2)
A(3)
A(4)
A(5)

要素の数＝6

2次元配列

DIM B$(2, 3)

B$(0, 0)	B$(0, 1)	B$(0, 2)	B$(0, 3)
B$(1, 0)	B$(1, 1)	B$(1, 2)	B$(1, 3)
B$(2, 0)	B$(2, 1)	B$(2, 2)	B$(2, 3)

要素の数　3×4＝12

各添字は原則として0から始まりますから，実際の要素の数は添字の最大値＋1となります．添字の最大値は0～32766までの範囲を整数形式で指定します．ただし，メモリによって添字の最大値および次元は制限されます．

また，添字の最大値が10以下のときは，DIM文による宣言を省略できます．この場合，添字の最大値は10とみなされます．

要素の参照は，変数名と各次元の添字を指定することによって行います．

例）
```
10 DIM D(3)
20 FOR I=0 TO 3
30 D(I)=I          ←―― 要素を格納
40 NEXT
50 PRINT D(2)  ←―― 添字が2の場合の要素を参照
60 END
```

1.7.4 予約変数

N_{88}-BASIC/N_{88}-日本語BASICには，BASIC自身が専用に使う予約変数があります．これらの変数は，ユーザによって一般の変数として使うことはできません．

TIME$　　現在の時分秒をHH:MM:SSの形で持っています．また時分秒を代入することもできます．

DATE$　　現在の年月日をYY/MM/DDの形で持っています．また年月日を代入することもできます．

ERL　　エラーが生じたとき，エラーが発生した行番号を持っています．代入することはできません．

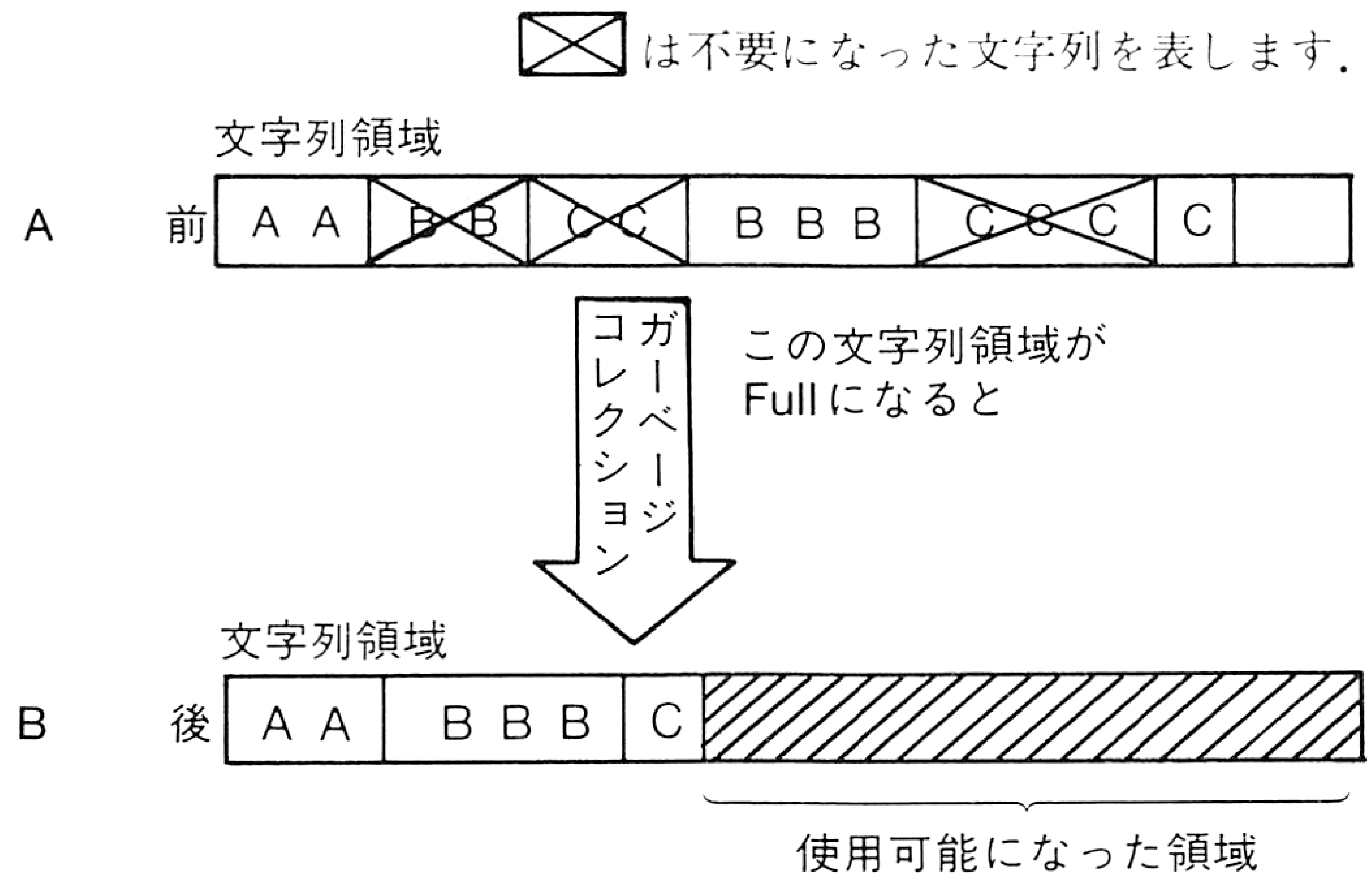
ERR　　エラーが発生したとき，生じたエラーのエラーコードを持っています．代入することはできません．

1.7.5 ガーベージコレクション

一般的に，文字列領域内には文字列が順番に存在するものではなく，あちこちに存在していて間には不要になった文字列があります．（下図A）そして，次々に文字変数に文字列が与えられ，文字列領域がFullの状態になると，現在使われている文字列の領域と不要になった文字列の領域の2つに分けて，使用可能な領域を確保する（下図B）ことをガーベージコレクションと言います．

このガーベージコレクションは，プログラムを実行しているときなど自動的に行われ，場合によっては処理に時間がかかることがあります．

概念図



文字列を多く扱うプログラムを実行中に，本機が一時的にストップした状態になるのはこれが原因です．

1.8　型変換

N_{88}-BASIC/N_{88}-日本語BASICの数値データは，必要に応じてその型から他の型に変換することができます．ただし，文字型と数値型の間でこの変換を行うことはできません．

(1)　ある型の数値データが，違った型の数値変数に代入された場合，数値はその変数名によって宣言された型に変換されます．

例)
```
10 ABC%=1.234   ←――実数を整数に変換
20 PRINT ABC%
RUN
 1
```

次は，実数を単精度型の数値変数に代入（単精度型に変換）したにもかかわらず，整数型の数値変数を指定したため0が表示された例です．

例)
```
10 ABC=0.123456789
20 PRINT ABC%
RUN
 0
```

このように，違った型の数値変数に代入されたものを参照したり，あるいは表示したりする場合は注意が必要です．

(2)　精度の違う数値間の演算の場合，精度の高い方に変換されて演算が行われます．たとえば，10#/3の場合は，10#/3#として演算が行われます．

例)
```
10 A#=10#/3
20 B#=10#/3#
30 PRINT A#,B#
RUN
 3.333333333333333    3.333333333333333
```

(3)　論理演算の場合，扱われる数値はすべて整数に変換され，結果は整数で与えられます．

例)
```
PRINT 12.34,NOT 12.34
 12.34          -13
```

＊NOT……否定を表します（1.9.3　論理演算参照）．

⑷　実数が整数に変換される場合は，小数点以下は四捨五入されます．このとき，整数型で扱える範囲を超えた場合はエラーが起こります．

例）

```
10 A%=34.4
20 B%=34.5
30 PRINT A%, B%
RUN
 34          35
```

```
10 A#=1.234E+07
20 B%=A#
30 PRINT B%, A#
RUN
Overflow in 20
```

⑸　倍精度変数が単精度変数に代入されたときは，変数の値は有効数字７桁に丸めたものとなります．単精度変数の精度は７桁であり，もとの倍精度の数値との誤差の絶対値は，5.96E−8以下となります．

例）

```
10 A#=1.23456789#
20 B!=A#
30 PRINT A#, B!
RUN
 1.23456789         1.23457
```

1.9　式と演算

式は定数や変数を演算子で結合して表します．また式の演算結果は，１つの数値あるいは１つの文字列になりますから，単に文字や数値あるいは変数だけのものも式になります．

例）

```
"BASIC"
3.14
10+3/5
A+B/C-D
TAN (DO)
```

BASICの演算は次の５つに分類されます．

1. 算術演算
2. 関係演算
3. 論理演算
4. 関　　数
5. 文字列演算

1.9.1　算術演算

算術演算子には次のようなものがあります．

実行順序	演算子	演算	例
↓	^	指数演算	X ^ Y
	−	負号	−X
	＊，／	乗算，実数の除算	X＊Y，X／Y
	＋，−	加算，減算	X＋Y，X−Y

演算の実行順序を変更する場合カッコを使用します．カッコの中の演算子は他の演算子より先に実行されます．カッコ内においては通常の実行順序に従います．

次に実行例を示します．

	代数表記	BASICの表記
(1)	$2X+Y$	2＊X+Y
(2)	$\frac{X}{Y}+2$	X/Y+2
(3)	$\frac{X+Y}{2}$	(X+Y)/2
(4)	X^2+2X+1	X ^ 2+2＊X+1
(5)	X^{Y2}	X ^ (Y ^ 2)
(6)	$(X^Y)^2$	X ^ Y ^ 2
(7)	$Y(-X)$	Y＊(−X)

(1)　整数の除算と剰余の計算

整数の除算は¥によって行われます．扱われる数値が実数の場合は，演算が実行される前に小数点以下が四捨五入されます．商は小数点以下が切り捨てられた整数となります．

例）　10¥3=3　　(10/3=3...余り1)

　　23.75¥5=4　　(24/5=4...余り4)

剰余の計算はMODによって行われます．扱われる数値が実数の場合は，演算が実行される前に小数点以下が四捨五入されます．結果は整数の割り算の余りです．

例）　13.3 MOD 4=1　　(13/4=3...余り1)

　　25.68 MOD 6.99=5　　(26/7=3...余り5)

(2)　0での除算

式の実行時に0での除算が行われた場合は，エラーメッセージを出力しますが，結果は計算機が扱うことのできる最大の数とみなし，それを除算の結果として処理を続行します．

また，べき乗の実行時に，0に対して負のべき乗を行った場合も同様になります．

```
例)     PRINT 2/0
        Division by zero
         1.70141E+38
        PRINT 0^-1
        Division by zero
         1.70141E+38
```

(3) **桁あふれ（オーバーフロー）**

代入や演算の結果がその変数の型内で表現することのできる範囲を超えた場合，桁あふれが発生します．

桁あふれが起こった場合，BASICは"Overflow"エラーを出力し，BASICが扱うことのできる最大の数を結果として与え，処理を続行します．

```
例)     PRINT 3^300
        Overflow
         1.70141E+38
```

1.9.2 関係演算

関係演算子は2つの数値を比較するときに用います．結果は，真(−1)，偽(0)で得られ，条件判定文などプログラムの流れを変えるのに用いられます(2章 IF…THEN…ELSE参照)．

関係演算子	内容	例
=	等しい	X=Y
< >，> <	等しくない	X< >Y，X> <Y
<	小さい	X<Y
>	大きい	X>Y
<=，=<	小さいか等しい	X<=Y，X=<Y
>=，=>	大きいか等しい	X>=Y，X=>Y

注意：=は代入文にも使うので注意すること．

IF文の中での使い方の例を次に示します．

```
IF X=0 THEN 1000
IF A+B< >0 THEN X=X+1:Y=Y+1
```

1.9.3　論理演算

論理演算子は複数の条件を調べたり，ビット操作や論理演算を行ったりするのに用います．論理演算は，ビットごとに0または1を結果として与えます．

各論理演算の内容を次に示します．

NOT　＝　not（否定）

X	NOT　X
1	0
0	1

AND　＝　and（論理積）

X	Y	X　AND　Y
1	1	1
1	0	0
0	1	0
0	0	0

OR　＝　inclusive or（論理和）

X	Y	X　OR　Y
1	1	1
1	0	1
0	1	1
0	0	0

XOR　＝　exclusive or（排他的論理和）

X	Y	X　XOR　Y
1	1	0
1	0	1
0	1	1
0	0	0

IMP　＝　implication（包含）

X	Y	X　IMP　Y
1	1	1
1	0	0
0	1	1
0	0	1

EQV　=　equivalence（同値）

X	Y	X EQV Y
1	1	1
1	0	0
0	1	0
0	0	1

論理演算子も関係演算子のように，プログラムの流れを変えるのに用いられます．この場合，論理演算子は2つ以上の関係演算子と結ぶことができます．

例）
(1) IF X<0 OR 99<X THEN 1000
(2) IF 0<X AND X<100 THEN X=0
(3) IF NOT (A=0) THEN 20

(1) Xが負，または99より大きければ，行番号1000へ飛ぶ．
(2) Xが正で，かつ100より小さければ，Xに0を代入する．
(3) Aが0でなければ，行番号20へ飛ぶ．

注意：論理演算子は演算の前に扱う数値を，−32768から+32767までの2つの補数表示の整数に変換します．もし，この範囲外となれば"Overflow"エラーとなります．もし，0（偽）と−1（真）しか与えられなかったなら，論理演算子は0と−1しか結果として与えません．

指定された論理演算では，この整数に対しビットごとに演算を行います．したがって，論理演算子はバイトデータをあるビットパターンに照らし合わせて調べることができます．

たとえば，AND演算子は機器のI/Oポートのステータスバイトの必要なビット以外のすべてのビットをマスクすることに使えます．またOR演算子はある2進数を作るために，2つのビットパターンを混在させることができます．

以下の例は論理演算子がどのように働くかの例です．

NOT　2=−3	$2=(0000000000000010)_2$
	したがって，NOT　$2=(1111111111111101)_2=-3$
NOT X=−(X+1)	任意の数の補数表示は1の補数に1を加えたものです．
−1　AND　3=3	$-1=(1111111111111111)_2$，$3=(0011)_2$
	したがって，−1　AND　$3=(0000000000000011)_2=3$
4　OR　3=7	$4=(0100)_2$，$3=(0011)_2$
	したがって，4　OR　$5=(0111)_2=7$
−2　XOR　4=−6	$-2=(1111111111111110)_2$，$4=(0100)_2$
	したがって，−2　XOR　$4=(1111111111111010)_2=-6$

−2　IMP　−3=−3　−2=$(1111111111111110)_2$，−3=$(1111111111111101)_2$

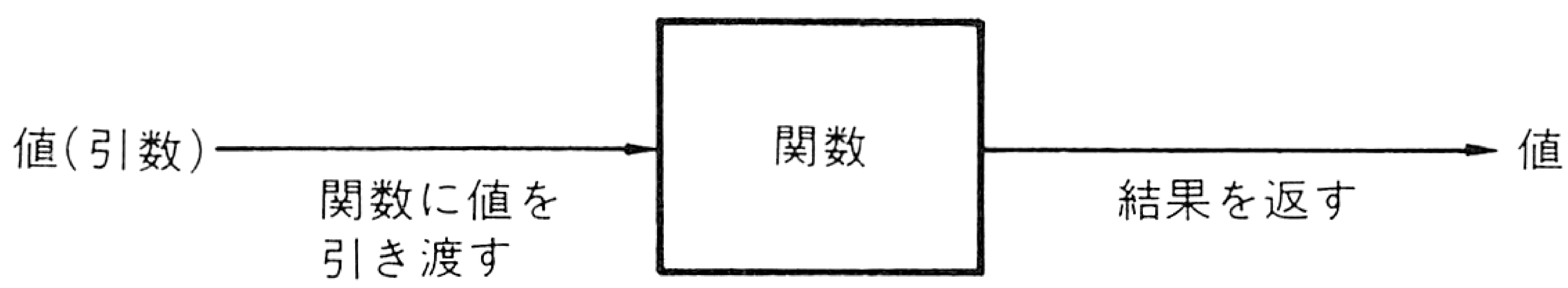
したがって，−2　IMP　−3=$(1111111111111101)_2$=−3

5　EQV　−4=6　　5=$(0101)_2$，−4=$(1111111111111100)_2$

したがって，5　EQV　−4=$(0000000000000110)_2$=6

1.9.4 関　数

ある値(引数)を引き渡して，決まった演算処理を行った結果を値として返してくるものが関数です．図で表すと次のようになります．



関数は単独で使われることはなく，PRINT文や代入文などで引用されます．

例)　　PRINT SIN(PI/2)

　　　　NUM$=MID$(STR$(NUMBER),2)

BASICでは“組み込み関数”としてSIN(正弦)，SQR(平方根)などの数値関数やCHR$, LEFT$などの文字列関数を本体内に持っています．

また，BASICは“ユーザ定義関数”としてユーザが自由に定義できる関数機能も持っています．これは2章の“DEF FN”の項で説明します．

また，これらの初等関数(SIN関数など)は，引数が倍精度のときは倍精度となり，引数が整数や単精度のときは単精度となります．

一般に引数に整数しかとらないものは，小数部分を四捨五入して整数に丸めてから演算を行います．

次にその使い方の例を示します．

```
A=SIN(3.14)+COS(3.14)
PRINT 2,2*2,SQR(2)
```

1.10 文字列の演算

1.10.1 文字列の連結

文字列は演算子＋によって連結することができます．

例）
```
10 A$=" 日本電気 ":B$=" Personal ":C$=" Computer "
20 D$=A$+" "+B$+" "+C$
30 PRINT D$
RUN
日本電気 Personal Computer
```

1.10.2 文字列の比較

文字も数値の比較に用いられるものと，全く同じ関係演算子を用いて比較することができます．

＝，＜，＞，＜＞，＞＜，＜＝，＝＜，＞＝，＝＞

文字列の場合，それぞれの文字列の最初から1文字ずつ文字の比較を行います．もし，相互が全く同じ文字列の場合は，その2つの文字列は等しくなりますが1箇所でも違った場合は，その文字のキャラクタコード（日本語文字の場合はシフトJISコード）の大きい方の文字列が大きくなります．キャラクタコードの大小は，英字のときはアルファベット順，カタカナのときはアイウエオ順になっています．文字列の片方が短かくて比較が途中で終わった場合は，短かい文字列の方が小さくなります．

参照：資料4 「キャラクタコード」，資料8 「日本語コード」

文字列の比較においては空白なども意味を持ちますから注意してください．

例）
```
"AA"<"AB"
"BASIC"="BASIC"
"PEN□">"PEN"
"cm">"CM"
"左"<"右"
```

1バイト文字（キャラクタコード表に載っている文字）と，2バイト文字（シフトJISコードで表される日本語文字）とが混在する文字列の比較は，2バイト文字を16進数の上2桁，下2桁に分割して1バイトずつ比較します．

例） "2月"<"二月" （32H, 8CH, 8EHと93H, F3H, 8CH, 8EHとの比較）

このように，文字列の比較は文字列の内容を調べたり，文字をアルファベット順に並べたり（ソート）することに使うことができます．

1.11　演算の優先順位

演算は次の順位によって行われます.

1. カッコで囲まれたもの
2. 関数
3. 指数(べき乗)^
4. 負号(—)
5. ＊，／
6. ¥
7. MOD
8. ＋，－
9. 関係演算子(＜，＞，＝など)
10. NOT
11. AND
12. OR
13. XOR
14. IMP
15. EQV

1.12　数値演算における誤差

通常の演算方式でプログラムを組んだところ，どうしても正しい答えが表示されないときに参照してください.

1.12.1　誤　　差

コンピュータの計算が紙の上での計算と全く同じと考えている方が大多数だと思われますが，次のように必ずしも同じであるとはいえないことがあります.

例)　　1. 実数型変数のプログラム

```
100 C=0:A=0
110 PRINT "C=";C,"A=";A
120 IF A=1 THEN 150
130 C=C+1:A=A+0.1
```

```
140 GOTO 110
150 END
```

このプログラムはカウントCと変数Aを0にしてから，変数Aが1となったところで終わるものです．カウントCは1ずつ，変数Aは0.1ずつ増えていきます．このプログラムを実行してみましょう．

```
RUN
C=0          A=0
C=1          A=0.1
C=2          A=0.2
C=3          A=0.3
C=4          A=0.4
C=5          A=0.5
C=6          A=0.6
C=7          A=0.7
C=8          A=0.8
C=9          A=0.9
C=10         A=1
C=11         A=1.1
C=12         A=1.2
C=13         A=1.3
```

不思議なことに，A=1.0となってもプログラムは止まりません．

例）　2. 関数のプログラム

```
100 S=15
110 FOR Y=0 TO S
120 A=2^Y
130 PRINT "2^";Y;"=";A
140 NEXT Y
150 END
```

このプログラムは2^0から2^{15}までを計算してプリントするものです．実行させてみましょう．

```
RUN
2^0=1
2^1=2
2^2=4
```

```
2^3=8
2^4=16
2^5=32
2^6=64
2^7=128
2^8=256
2^9=512
2^10=1024
2^11=2048
2^12=4096
2^13=8192.01
2^14=16384
2^15=32768
```

ここで，2^13に注目してください．8192.01というように，小数点以下2桁まで表示されていますね．

例） 3. 数値表示プログラム

```
100 A#=0.0075 (注)
110 PRINT A#
120 END
```

注意：#は変数Aが倍精度変数として扱われることを表しています．

このプログラムを実行してみましょう．

```
RUN
 7.500000298023224D-03
```

例）1のプログラムでは，A=1.0となっているのにプログラムが止まらず，例）2と例）3のプログラムでは，予想に反した結果が表示されます．その理由を簡単に説明すると次のようになります．

例）1のプログラムでは，A=1.0と表示されたときに，実はちょうど1になっていないからです．この現象は，コンピュータにおける数値の内部表現形式が原因となって起こります．後でもう少し詳しく説明します．

例）2のプログラムで小数点が現れたのは，関数の値を近似計算式で求めているためです．そして，他の関数(SIN, COS, TAN, SQR, etc.)についても，同じように近似計算を行っているための誤差が含まれて出力されます．

例）3の場合は，データを内部表現形式に変換する場合に生まれる誤差の影響です．

1.12.2 対　策

それでは，どうすれば誤差の影響の少ないプログラムになるでしょうか．主な対策をあげてみましょう．

a.　実数型変数を比較する場合は等号(＝)ではなく，不等号(＞,＜)を使用します．

b.　FOR～NEXTループの変数には整数型変数を使用します．

c.　結果を整数値として表示する場合は，INT関数やFIX関数を使用します．このとき注意しなければならないのは，理論上，結果が

```
A=1
```

となるものが

```
A=0.9999……
```

となっている場合です．このとき，このままでINT関数を使用すると，

```
PRINT INT(A)
  0
```

となり，結果が0になってしまうときがあります．これは，INT関数が与えられた引数の整数部分だけを取り出すからです．このような場合は，INT関数に代入する前に0.1を加えておくという手段が有効です．

d.　結果を画面やプリンタに出力する場合は，PRINT USING文を利用します．

例1の場合には，aに従い行番号120の文を次のように変更します．これで小数点以下4桁の精度を必要とする場合の比較を行うことができます．

単精度の数値は有効桁6桁，倍精度は有効桁16桁ですので，このことを考慮して精度を決定してください．

```
120 IF 0.9999<A AND A<1.0001 THEN 150
```

1.12.3 数値の内部表現形式

一般にコンピュータ内部での数値の演算方式には次の2通りの方式があります．

2進演算方式……入力された数値を計算機内部で2進数に変換した後，いろいろな演算を行う．

例)　　入力値　$123_{(10)}$

　　　　内部値　$1111011_{(2)}$

10進演算方式……入力された数値を10進数のまま計算機内部に記憶し，演算を行う．

例)　　入力値　$123_{(10)}$

　　　　内部値　00000001　00000010　$00000011_{(2)}$

　　　　　　　　　　　↓

　　　　　　　　　$1\ 2\ 3_{(10)}$

注意：$_{(2)}$は2進数を表し，$_{(10)}$は10進数であることを表しています．

これら2つの演算方式の大きな違いは，数値を2進数に変換するところです．つまり，10進数の形で残すか，2進数の形に変換してしまうかによって，再び10進数として表示する場合に誤差が生まれるかどうかが決まってきます．

たとえば，0.83を2進演算用に変換すると次のようになります．

$$0.83=1/2^1+1/2^2+1/2^4+1/2^6+1/2^{10}+1/2^{11}+1/2^{12}+\cdots\cdots$$

したがって，計算機内部では，

1101010001111……

という形になります．しかし，計算機のメモリには限度があり，通常は32ビットとか64ビットというように制限されています．したがって，再び10進数に変換すると前とは異なった値になります．

ですから，2進演算方式を採用している本機では多少の誤差が現れますが，これは特に異常ということではありません．

では，なぜ多くのパーソナルコンピュータが2進演算を採用しているのでしょうか．主な理由は次の2つです．

1. 演算を早く行う．
2. 大きな数値でも少ない桁数で表現できる．

例) $255_{(10)}$

10進演算

00000010　00000101　00000101

（3バイト）

2進演算

11111111

（1バイト）

1.12.4 全角文字の内部形式

全角文字は，N_{88}-日本語BASIC使用時は，文字列の中で半角文字と混在して扱うこともできます．

全角文字1文字は，半角文字2文字分のメモリ（2バイト）を占有します．

例) "ABC漢字DE"の内部形式(A, B, C, D, Eは半角文字)

<table>
<tr><td>A</td><td>B</td><td>C</td><td colspan="2">漢</td><td colspan="2">字</td><td>D</td><td>E</td><td>内部形式</td></tr>
<tr><td>41</td><td>42</td><td>43</td><td>8A</td><td>BF</td><td>8E</td><td>9A</td><td>44</td><td>45</td><td>16進コード</td></tr>
</table>

1.13　基本命令と拡張命令

1.13.1　基本命令

本機で使用するN_{88}-BASIC/N_{88}日本語BASICのすべての命令のうち，11個の拡張命令を除いた命令を基本命令といいます．基本命令は，その機能からコマンド，ステートメント，関数などに分類することができます．（詳しくは本書vii～xviiiページ「機能別索引」を参照してください．）

第2章では，これらの基本命令をアルファベット順に解説してあります．

1.13.2　拡張命令

N_{88}BASIC V2/N_{88}-日本語BASICでは，基本命令のほかに拡張命令を追加することができます．N_{88}-BASIC V1では追加することができません．拡張命令は，NEW CMD文を実行することによって追加されます．これにより，512色のうちから8色を選べるアナログパレット機能やFM音源によるサウンド機能を使うことができます．

2章・3章の見方

N₈₈-BASICおよびN₈₈-日本語BASICのすべてのコマンド，ステートメント，関数，予約変数を2章・3章で解説します．BLOADというステートメントを解説したページを例にあげて，使われている記号や表記方法を説明します．

表記例

② ①

入出力コマンド

BLOAD ディスク

ビー・ロード：binary load

③ ④

⑤→ **機能** 機械語プログラムをロードする

⑥→ **書式** BLOAD〈ファイル名〉〔,〈ロードアドレス〉〕〔,R〕

⑦→ **解説**

• 〈ファイル名〉によって指定された機械語プログラムファイルをメモリにロードします．このファイルは，BSAVEコマンドで作成されたものでなければなりません．

• 〈ロードアドレス〉が省略された場合，BSAVEコマンドで指定した番地からロードされます．また，〈ロードアドレス〉が指定されるとその番地からロードされます．

• Rオプションを指定するとプログラムロード後，セーブの際に指定された〈セーブアドレス〉から，プログラムを実行します．このとき，すでに開かれているファイルはその状態を保持します．〈ロードアドレス〉が指定されている場合は，〈ロードアドレス〉からプログラムを実行します．

• ただし，&H8000～&H83FFおよび&HE600～&HFFFFの領域でBLOADすることはできません．また，N₈₈-BASIC V2でNEW CMD文を実行した場合は，&HE100～&HE5FFの領域でもBLOADすることはできなくなります．

• BLOADを実行する前には，データを転送する領域をCLEAR文で確保してください．CLEAR文を実行しない場合は正常に動作しません．

⑧→ **例** BLOAD "demo.bin", &HD000

• "demo.bin"というファイル名の機械語ファイルを，D000H番地からロードします．

⑨→ **実行例** ディスク ←⑩

```
clear ,&hdfff⏎
Ok
bload "2:test.m",&he000⏎
Ok
```

• E000H番地から機械語プログラムをロードします．

• ドライブ2に入っているフロッピーディスクの中の"test.m"というファイルを，E000H番地からロードします．BLOADコマンドの前のCLEAR文で機械語プログラム領域を確保しています．

⑪→ 注意：• カセットテープを対象としたBLOADコマンドは実行できません．

⑫→ 参照：BSAVE

①機能別分類

BASIC上における機能を示します．その機能を大きく分けると次の4つに分類されます．

(1) コマンド

プログラムの実行や編集などを指示する命令です．一般的にダイレクトモードで使用し，実行後"Ok"が表示されます．

(2) ステートメント

単独で，ある動作を行う命令です．主にプログラム中で使用します．(コマンドとステートメントは必ずしも明確に分かれていません．)

(3) 関数

ある値(引数)を引き渡して処理された結果を，値として返す機能を持ちます．単独で使われることはなく，代入文やコマンド，ステートメント中で使われます．

(4) 予約変数

BASICが専用に使っている変数です．

さらにこのマニュアル中では，これらの機能を次のように分類しています．

- **一般コマンド，一般ステートメント**

  プログラムを作る，動作させる，停止させる，変数などの定義をするといった基本的な動作や，プログラムの流れのコントロール(繰り返しや分岐)を行います．

- **入出力コマンド・入出力ステートメント・入出力関数**

  フロッピーディスクユニット，カセットテープレコーダ，画面，キーボード，プリンタ，スピーカ，RS-232C通信ポートなどの操作を行います．

- **画面制御ステートメント，画面制御関数，画面制御予約変数**

  テキスト画面およびグラフィック画面の初期化やテキスト画面に対する操作を行います．

- **グラフィックステートメント，グラフィック関数**

  グラフィック画面に対する操作を行います．

- **数値関数**

  数値演算を行います．

- **文字ステートメント，文字関数，文字予約変数**

  文字列の操作を行います．

- **特殊コマンド・特殊ステートメント・特殊関数・特殊予約変数**

  機械語モニタに入るとき，エラー処理，機械語や入出力ポートの操作などの特殊な働きをします．

- **日本語ステートメント，日本語関数**

  日本語文字に対する操作を行います．

• **拡張ステートメント**

拡張命令を追加して使用するサウンド機能に対する操作を行います.

② ディスク このマークがついている命令はDISK version(N_{88}-BASICシステムディスク，またはN_{88}-日本語BASICシステムディスク使用)でサポートされています．したがって，ROM versionで使うことはできません.("Disk BASIC Feature"エラーとなります.)

N88-日本語 このマークがついている命令はN_{88}-日本語BASICでサポートされています(N_{88}-日本語BASICシステムディスク使用)．したがって，N_{88}-BASICで使うことはできません.

このマークのついていない命令，関数で，N_{88}-日本語BASIC固有の事柄については，文章中にアミ(　　)をかけてわかりやすくしてあります.

これらのマークのついていないものは，N_{88}-BASIC ROM version，N_{88}-BASIC DISK version，およびN_{88}-日本語BASICのいずれでも使うことができます.

③ 命令の読み方を表します.

④ 命令をフルネームで示します.

⑤ **機　能** 命令の機能を簡単に説明します.

⑥ **書　式** 命令の記述の仕方を示します．実際に入力を行う場合は次のような決まりに従ってください.

1　カギカッコ〈，〉に囲まれていない文字や記号はそのまま入力します．文字は大文字でも小文字でもかまいません.

ただし，次の場合は大文字と小文字を区別しなければなりません.

(1)　ダブルクォーテーション(")で囲まれた文字列は大文字と小文字を区別します.

例)　"dskut2.j88" ←――ファイル名

set 1,"P" ←――属性文字

(2)　数値を指数形式で表す場合に用いるEとDは大文字でなければなりません.

例)　A!=−7.09E−06

B#=1.094328564D5

2　カギカッコ〈,〉で囲まれた項目は，ユーザが具体的に指定します．項目の主な種類は次のとおりです.

〈数式〉:　数値定数や数値変数を単独で，あるいは演算子で結合したものです.

例)　1, 10, 1/SIN(A!), B2, C＊D

〈文字列〉:　文字定数あるいは文字型変数を表します.

例)　"ABC", ST$

〈**式**〉：　〈数式〉または〈文字列〉を表します．

〈**機能**〉，〈**スイッチ**〉，〈**コード**〉：

0，1，2，3などの整数で，コマンドやステートメントの機能を指定するものです．

〈**ファイルディスクリプタ**〉：

"デバイス名　ファイル名"の形式を持ちます．ファイル名はさらにファイル名，拡張子の書式を持ちます．(詳しい説明は，**プログラマーズガイド**　第13章　「入出力装置の管理」を参照してください．)

例)　"2：dskut2．j88"，"demo1"

〈**ファイル番号**〉：ファイルをオープンするときに定義します．BASICを起動したときに設定した数の範囲内で指定します．

〈**ドライブ番号**〉：フロッピーディスクユニットのおのおののドライブに割り当てられる番号です．

PC-8801MHの場合

5.25インチミニフロッピーディスクユニット		8インチフロッピーディスクユニット			
本体内蔵フロッピーディスクドライブ		PC-8881 PC-8881 (1)		PC-8882 PC-8882 (1)	
drive 1	drive 2	drive 1	drive 2	drive 3	drive 4
1	2				
3	4	1	2		
5	6	1	2	3	4

表中の1～6までの番号が割り当てられるドライブ番号です．

空白は，フロッピーディスクユニットが接続されていないことを表します．

PC-8801FHの場合

5.25インチフロッピーディスクユニット				8インチフロッピーディスクユニット			
PC-80S31K PC-80S31 PC-8031-2W PC-8031-1W		PC-80S32 PC-8032-2W PC-8032-1W		PC-8881 PC-8881(1)		PC-8882 PC-8882(1)	
drive 1	drive 2	drive 3	drive 4	drive 1	drive 2	drive 3	drive 4
1	2						
1	2	3	4				
				1	2		
				1	2	3	4
3	4			1	2		
3	4	5	6	1	2		
5	6			1	2	3	4
5	6	7	8	1	2	3	4

表中の1～8までの番号が割り当てられるドライブ番号です．

空白は，フロッピーディスクユニットが接続されていないことを表します．

〈行番号〉：　行番号の他にラベル名を含みます．

〈カラーコード〉：画面に表示可能な8色の色につけられた番号です．

0：黒　1：青　2：赤　3：紫

4：緑　5：水色　6：黄色　7：白

3　角カッコ〔，〕で囲まれた項目は省略することができます．省略した場合，デフォルト値(BASICによって設定される値)または以前に指定した値が適用されます．

4　上記のカギカッコ，角カッコ以外の記号でカッコ(　)，コンマ(，)，セミコロン(；)，ハイフン(－)，等号(＝)などの記号は，示された位置に正しく入力します．

5　省略記号(...)の続く項目は，1行の許す長さ(255文字)内で繰り返すことができます．

例)　〈定数〉〔,〈定数〉...〕の場合　0,10,15

6　座標指定のところで(Wx, Wy)と記されているのはワールド座標を，(Sx, Sy)はスクリーン座標を表しています．その他，単に(X, Y)と記されているのはキャラクタ座標を表しています．また，STEP(x. y)という記法は，相対座標による指定ができることを意味しています．

⑦ **解　説**　命令の使用方法や詳しい機能，それに関して注意しなければならない点などを説明します．

⑧ **例** 実際の入力の見本として簡単な例とそれについての解説を示します．

命令が関数の場合は，**返される値** という記号を使って処理された結果も同時に示します．

⑨ プログラム例 または 実　行　例

プログラム例には，その命令といくつかの文を交えたプログラムとその実行結果，および解説が示されています．実行例には，その命令の実際の使い方と解説が示されています．

アミ（　）の部分はユーザが実際に入力する部分で，記述されているとおりに入力した場合は，続く実行結果と同じになります．ただし，システムの構成が違う場合など必ずしも同じ実行結果にならないことがあります．

注意：（N88-日本語 マーク）マークのついていない命令，関数のサンプルプログラムは，原則として，N88-BASIC，N88-日本語BASICのどちらでも実行することができます．ただし，特定のメモリ領域を使用するもの(CALL, DEF USRなど)や，グラフィックシンボルを表示するものなどは，N88-日本語BASICでは実行できないものがあります．このような場合は，アドレスをずらしたり，プログラムの最初にSCREEN文を実行して，グラフィックシンボルモードにしておく，などの変更が必要になります．

サンプルプログラム中に，全角文字が含まれているものは，N88-日本語BASICの日本語モード（本文 P2-209 SCREEN文参照）でプログラムを入力した場合です．

N88-BASICでサンプルプログラムを実行するときは，入力するときに全角文字の部分を半角の他の文字列に置き換えれば，同じように実行することができます．

下の例を参考にしてください．

プログラム例

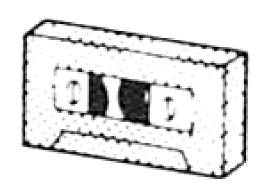
```
100 ' ABS ｻﾝﾌﾟﾙ
110 INPUT "点１の座標は：　Ｘ，　Ｙ　";X1,Y1
120 INPUT "点２の座標は：　Ｘ，　Ｙ　";X2,Y2
130 D=SQR(ABS(X1-X2)^2+ABS(Y1-Y2)^2)
140 PRINT:PRINT "点１と点２の距離は　";D
150 END
Ok
```

上記プログラム例の入力では，100行，120行と140行においてアンダーラインを引いた部分の文字列が全角文字です．その他の部分は半角文字です．N88-BASICをお使いの方は，アンダーラインの部分を半角文字のカタカナ，英数に置き換えてください．

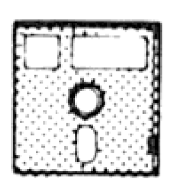
全角文字の入力の仕方は，**プログラマーズガイド**第５章「日本語文字の入力」を参照してください．

⑩　システム構成

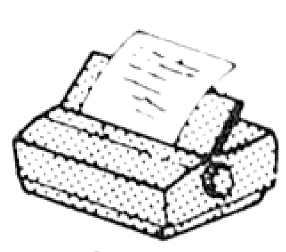
プログラム例あるいは実行例を実行する際，本体とディスプレイの他に必要な機器を示します．次のマークがついていない場合は，本体とディスプレイのみで実行できます．


カセットテープ

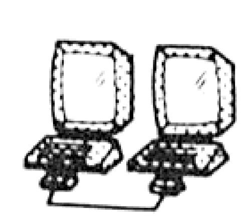
… CMTインタフェースボード（PC-8801MHの場合）
カセットテープレコーダ
カセットテープ


ディスク

… フロッピーディスクユニット
フロッピーディスク


プリンタ

… プリンタ
用紙


RS-232C

… RS-232Cに接続される機器(他のパーソナルコンピュータなど)
RS-232Cケーブル

⑪　**注意**：特に注意してほしい点を示します．

⑫　**参照**：関連の深い項目を示します．

第2章

基本命令

ABS

アブソリュート：absolute

機　能　絶対値を返す

書　式　ABS (〈数式〉)

解　説

• 〈数式〉の値の絶対値を返す関数です．

• ABS関数の演算結果は，〈数式〉に倍精度実数が含まれる場合は倍精度に，その他の場合は単精度になります．

例

ABS(−2.4)　返される値 ⇒　2.4

ABS(2.4)　返される値 ⇒　2.4

ABS(0)　返される値 ⇒　0

プログラム例

```
100 ' ABS ｻﾝﾌﾟﾙ
110 INPUT "点１の座標は：　Ｘ，Ｙ　";X1,Y1
120 INPUT "点２の座標は：　Ｘ，Ｙ　";X2,Y2
130 D=SQR(ABS(X1-X2)^2+ABS(Y1-Y2)^2)
140 PRINT:PRINT "点１と点２の距離は　";D
150 END
Ok
run⏎
点１の座標は：　Ｘ，Ｙ　？ 0,0⏎
点２の座標は：　Ｘ，Ｙ　？ 100,100⏎

点１と点２の距離は　 141.421
Ok
```

• 2点の座標を読み込んで，距離を出力します．

• 2点の座標を(X1, Y1), (X2, Y2)とすると距離は，

$$距離 = \sqrt{|X1-X2|^2 + |Y1-Y2|^2}$$

によって得られるので，プログラムでは130行のようになります．

AKCNV$

N88-日本語

エー・ケー・シー・エヌ・ブイ・ダラー：ascii kanji convert

機　能　文字列中の半角文字を全角文字に変換した文字列で返す

書　式　AKCNV$(〈文字列〉)

解　説　・〈文字列〉の中に含まれる半角文字を全角文字に変換します.

・変換する〈文字列〉および変換された結果が255バイトを超えた場合は，"String too long" エラーとなります.

例　AKCNV$("NEC日本電気") 返される値 ＮＥＣ日本電気

・文字列中の半角文字"NEC"を全角文字に変換します.

プログラム例

N88-日本語

```
100 ' AKCNV$ ｻﾝﾌﾟﾙ
110 A$="日本語BASIC"
120 PRINT A$
130 PRINT AKCNV$(A$)
140 END
Ok
run⏎
日本語BASIC
日本語ＢＡＳＩＣ
Ok
```

・110行で，"日本語BASIC"を文字列に代入します.

・120行で，半角文字の"BASIC"を半角文字のまま文字列を表示します.

・130行で，半角文字の"BASIC"を全角文字に変換した文字列を表示します.

参照：KACNV$，資料７「半角文字/全角文字コード変換表」

ASC

アスキー：ascii

機能 文字に対応するキャラクタコード，シフトJISコードを返す

書式 ASC(〈文字列〉)

解説
- 〈文字列〉の最初の文字を対応するキャラクタコード(数値)に変換します．
- N_{88}-日本語BASIC日本語モードでは，〈文字列〉の最初の文字(半角文字，全角文字)を対応する文字コード(半角文字の場合はキャラクタコード，全角文字の場合はシフトJISコード)に変換します．
- グラフィックシンボルモードでは，N_{88}-BASICと同じように動作します．
- 〈文字列〉にヌルストリングを与えたときは，"Illegal function call"エラーが起こります．

例

ASC("NEC") 返される値 78

ASC("亜") 返される値 —30561

ASC("") 返される値 "Illegal function call"

プログラム例

```
100 ' ASC ｻﾝﾌﾟﾙ
110 A$=INPUT$(1)
120 IF ASC(A$)=13 THEN END
130 PRINT A$;" のｷｬﾗｸﾀ ｺｰﾄﾞは";ASC(A$);" です。"
140 GOTO 110
Ok
run⏎
P のｷｬﾗｸﾀ ｺｰﾄﾞは 80  です。
C のｷｬﾗｸﾀ ｺｰﾄﾞは 67  です。
- のｷｬﾗｸﾀ ｺｰﾄﾞは 45  です。
8 のｷｬﾗｸﾀ ｺｰﾄﾞは 56  です。
8 のｷｬﾗｸﾀ ｺｰﾄﾞは 56  です。
0 のｷｬﾗｸﾀ ｺｰﾄﾞは 48  です。
1 のｷｬﾗｸﾀ ｺｰﾄﾞは 49  です。
Ok
```

- キーボードから1文字読み込み，そのキャラクタコードを出力します．
- 110行で，INPUT$関数を使って1文字読み込みます．
- 130行で，読み込んだ文字とその文字のキャラクタコードを出力します．実行例では"P"，"C"，"-"，"8"，"8"，"0"，"1"そして⏎を入力しています．⏎を押すと120行でプログラムが終了します(⏎のキャラクタコードは13です)．

参照：CHR$，資料4「キャラクタコード」

ATN

アーク・タンジェント：arc tangent

機　能　逆正接(アークタンジェント)を返す

書　式　ATN(〈数式〉)

解　説

- 〈数式〉の値の逆正接(アークタンジェント)を返します.
- 返される値の単位はラジアン(π/180×角度)で，返される値の範囲は－π/2からπ/2までです.
- ATN関数の演算は単精度で行われます.

例

ATN(1)　返される値　.785398

ATN(0)　返される値　0

プログラム例

```
100 ' ATN ｻﾝﾌﾟﾙ
110 INPUT "数値を入力してください（－1から1まで）";A
120 IF ABS(A)>1 THEN 110
130 IF ABS(A)=1 THEN AS=1.5708:AC=0:GOTO 160
140 AS=ATN(A/SQR(-A*A+1))
150 AC=-ATN(A/SQR(-A*A+1))+3.14159/2
160 PRINT "arcCOS(";A;")=";AC
170 PRINT "arcSIN(";A;")=";AS
180 END
Ok
run
数値を入力してください（－1から1まで）? .5
arcCOS( .5 )= 1.0472
arcSIN( .5 )= .523599
Ok
```

- －1から1までの数値を読み込んで，逆余弦，逆正弦の値を求めます.
- 140行と150行では，

$$\arcsin A = \arctan (A/\sqrt{1-A^2})$$

$$\arccos A = -\arctan (A/\sqrt{1-A^2}) + \frac{\pi}{2}$$

を使って計算しています.

参照：SIN, COS, TAN

ATTR$

ディスク

アトリビュート・ダラー：attribute $

機　能　ファイルまたはフロッピーディスクの属性を返す

書　式　ATTR$(〈ドライブ番号〉)

ATTR$(#〈ファイル番号〉)

ATTR$(〈ファイル名〉)

解　説

• 指定したファイル，フロッピーディスクの属性を返す関数で，与えられるパラメータによって次の値を返します．

〈ドライブ番号〉……指定したドライブに入っているフロッピーディスクの属性

〈ファイル番号〉……ファイル番号によってオープンされているファイルの属性

〈ファイル名〉………ファイル名で指定されたファイルの属性

• 属性は3文字の文字列で返され，次のような意味を持ちます．

"   "：属性は"なし"です．読み込みおよび書き込みが可能な状態です．

"R "：書き込みのときにリードアフターライト(書き込んだ内容とメモリ上の内容の比較チェック)を行います．

" E "：ファイルが暗号化されています.(ファイルのみ)

" P"：書き込みが禁止されています．

• 属性を表す文字列の2文字目は常に空白で意味を持ちません．

例

ATTR$(1)　返される値▶　(ドライブ1に入っているフロッピーディスクの属性)

ATTR$(#2)　返される値▶　(ファイル番号2でオープンしているファイルの属性)

ATTR$("demo . n88")　返される値▶　("demo . n88"というファイル名が付いたファイルの属性)

プログラム例

ディスク

```
100 ' ATTR$ サンプル
110 INPUT "ファイル名 : ";F$
120 AT$=ATTR$(F$)
130 PRINT "ファイル ";F$;" の属性は : ";
140 A$="解除"
150 IF AT$="R  " THEN A$="リード アフター ライト"
160 IF AT$="  P" THEN A$="ライト プロテクト"
170 PRINT A$+" です。"
180 END
Ok
run
ファイル名 : ? data
ファイル data の属性は : 解除 です。
Ok
```

・指定されたファイルの属性を出力します．

・120行で指定されたファイルの属性を表す文字を変数AT$に代入しています．

・140行から160行で変数AT$の内容に従って，"解除"，"リード　アフター　ライト"そして"ライト　プロテクト"のいずれかを変数A$に代入します．

参照：SET

AUTO

オート：auto

機能 行番号を自動的に発生させる

書式 AUTO〔〈行番号〉〕〔,〈増分〉〕

解説
- AUTOコマンドは，プログラム入力時に行番号を自動的に発生させるモードを指定します．
- 〈行番号〉は，最初に発生させる行番号で，以後，[⏎]の入力ごとに〈増分〉を加えた行番号を発生させます．
- 〈行番号〉は1～65528まで，〈増分〉は1～65527までの整数であり，指定のないときはそれぞれ10とみなされます．
- [コントロール] ＋ [C] または [停止] を押すと，AUTOモードから抜け出し，BASICのコマンドレベルに戻ります．このとき，最後に発生させた行番号の行は格納されません．
- プログラム中にすでに存在する行と同じ行番号が発生された場合には，行番号の直後にアスタリスク(＊)が表示され，注意を促します．このとき，文字を入力して[⏎]を押すと，その行は新しい内容に変わります．また，何も入力しないで[⏎]を押した場合には，その行は削除されます．
- AUTOのモード中においても，カーソルエディッテング機能を利用することができます．カーソルを移動してすでに入力した行，あるいは表示されていた行のところで，[⏎]を入力した場合には，その行の行番号に増分を足した行番号が次に発生される行番号となります．
- なお，AUTOコマンドでは〈行番号〉を指定する際，現在の行を表すピリオド(．)を指定することができます．

例

AUTO

- 10行から10行刻みで行番号を発生させるモードに入ります．

AUTO 1000,10

- 1000行から10行刻みで行番号を発生させるモードに入ります．

実 行 例

```
auto 100,20[⏎]
100 ' AUTO ｻﾝﾌﾟﾙ[⏎]
120 print "AUTO test"[⏎]
140 end[⏎]
160 [停止]
list[⏎]
100 ' AUTO ｻﾝﾌﾟﾙ
120 PRINT "AUTO test"
140 END
Ok
```

• 行番号を自動的に発生させてプログラムを入力します.

• この実行例では，100行から20行刻みに行番号を発生させています.

• 140行まで入力したら，160行が発生した時点で [停止] を押してコマンドレベルに戻ります.

BEEP

ビープ：beep

機能 ブザーを鳴らす

書式 BEEP〔〈スイッチ〉〕

解説

• 内蔵スピーカによりブザーを鳴らしたり，止めたりします．

• 〈スイッチ〉が1のときブザーは鳴り続け，〈スイッチ〉が0で止まります．

• 〈スイッチ〉を省略した場合には，一定時間ブザーを鳴らします．

これは，PRINT CHR$(7); を実行するのと同じです．

• 音量は，本体表側の内蔵スピーカ音量調節ボリュームによって調節できます．

例

BEEP

• 一定時間ブザーを鳴らします．

BEEP 0

• ブザーを止めます．

プログラム例

```
100 ' BEEP ｻﾝﾌﾟﾙ
110 FOR I=&H41 TO &H44
120   PRINT CHR$(I);" を入力してください。";
130   A$=INPUT$(1)
140   IF A$=CHR$(I) THEN PRINT A$ ELSE BEEP:GOTO 130
150 NEXT I
160 END
Ok
run
A を入力してください。A
B を入力してください。B
C を入力してください。C
D を入力してください。D
Ok
```

• 入力する文字を間違えたときブザーを鳴らします．

• "A"から"D"までの文字を順番に入力するようにメッセージを表示しますので，指定された文字を入力してください．違う文字を入力した場合はブザーが鳴り，正しい文字を入力するまで待ち続けます．

• 140行で入力した文字と指定した文字が同じであれば，その文字を出力し，同じでなければブザーが鳴り，130行の入力待ち状態に戻ります．

BLOAD

ディスク

ビー・ロード：binary load

機能　機械語プログラムをロードする

書式　BLOAD 〈ファイル名〉〔,〈ロードアドレス〉〕〔,R〕

解説　•〈ファイル名〉によって指定された機械語プログラムファイルをメモリにロードします．このファイルは，BSAVEコマンドで作成されたものでなければなりません．

•〈ロードアドレス〉が省略された場合，BSAVEコマンドで指定した番地からロードされます．また，〈ロードアドレス〉が指定されるとその番地からロードされます．

• Rオプションを指定するとプログラムロード後，セーブの際に指定された〈セーブアドレス〉から，プログラムを実行します．このとき，すでに開かれているファイルはその状態を保持します．〈ロードアドレス〉が指定されている場合は，〈ロードアドレス〉からプログラムを実行します．

•ただし，&H8000～&H83FFおよび&HE600～&HFFFFの領域でBLOADすることはできません．また，N_{88}-BASIC V2でNEW CMD文を実行した場合は，&HE100～&HE5FFの領域でもBLOADすることはできなくなります．

•BLOADを実行する前には，データを転送する領域をCLEAR文で確保してください．CLEAR文を実行しない場合は正常に動作しません．

例　BLOAD "demo.bin", &HD000

•"demo.bin"というファイル名の機械語ファイルを，D000H番地からロードします．

実行例

ディスク

```
clear,&hdfff⏎
Ok
bload "2:test.m",&he000⏎
Ok
```

•E000H番地から機械語プログラムをロードします．

•ドライブ2に入っているフロッピーディスクの中の"test.m"というファイルを，E000H番地からロードします．BLOADコマンドの前のCLEAR文で機械語プログラム領域を確保しています．

注意：•カセットテープを対象としたBLOADコマンドは実行できません．

参照：BSAVE

BSAVE

ディスク

ビー・セーブ：binary save

機能 機械語プログラムをセーブする

書式 BSAVE〈ファイル名〉,〈セーブアドレス〉,〈長さ〉

解説
- メモリ上に置かれている機械語プログラムを，指定された〈ファイル名〉でセーブします．
- 〈セーブアドレス〉で指定された番地から，〈長さ〉バイトの内容を機械語プログラムとしてセーブします．〈長さ〉は〈終了番地〉−〈開始番地〉＋１で求められます．
- 〈セーブアドレス〉を実行開始番地としておくと，BLOADコマンドにより，機械語プログラムのロードと同時に実行を開始することができます．ただし，&H8000〜&H83FFの領域をBSAVEすることはできません．

例 BSAVE "demo.bin", &HE000, 500

- E000H番地から500バイトの機械語プログラムを，"demo.bin"というファイル名でセーブします．

実行例

ディスク

```
bsave "2:test.m",&he000,&h100⏎
Ok
```

- E000H番地からE0FFH番地までの内容を，"test.m"というファイル名でドライブ２に入っているフロッピーディスクにセーブします．

注意：• カセットを対象としたBSAVEコマンドは実行できません．

参照：BLOAD

CALL

コール：call

機能 機械語サブルーチンを呼び出す

書式 CALL〈変数名〉〔(〈引数〉〔,〈引数〉…〕)〕

解説

- メモリ中に用意された機械語サブルーチンに制御を移します．
- 〈変数名〉は機械語サブルーチンの実行開始番地を指定し，〈変数名〉で指定されている値が実行開始番地となります．
- 〈変数名〉には配列変数を用いることはできません．
- 〈引数〉は，機械語サブルーチンに渡す変数を指定します．
- 引数としてはすべての型の変数を指定することができますが，定数や式を渡すことはできません．
- CALL文によって呼び出されるサブルーチンは，機械語のRET命令によりBASICに戻すことができます．

例 CALL MSUBR(ARG1,ARG2)

- 変数MSUBRで指定された番地から，機械語プログラムを実行します．
- "ARG1", "ARG2"で指定された変数を，機械語サブルーチンに渡します．

プログラム例

- このプログラムは，N_{88}-BASICで実行してください．

```
100 ' CALL ｻﾝﾌﾟﾙ
110 CLEAR,&HDFFF
120 GOSUB 210
130 CONSOLE 0,25,0,1:WIDTH 80,25:CLS
140 FOR I=1 TO 80*24
150   PRINT "♥";
160 NEXT I
170 CLS:FOR I=1 TO 500:NEXT I
180 VAD%=&HF3C8:AC=&HE000
190 CALL AC(VAD%)
200 END
210 FOR AD=&HE000 TO &HE015
220   READ DA:POKE AD,DA
230 NEXT AD
240 RETURN
250 '
260 DATA &H7E           :'      LD   A,(HL)
270 DATA &H23           :'      INC  HL
280 DATA &H66           :'      LD   H,(HL)
290 DATA &H6F           :'      LD   L,A
300 DATA &H3E,&HE9      :'      LD   A,"♥"
310 DATA &H11,&H28,&H00 :'      LD   DE,40
320 DATA &H0E,&H19      :'      LD   C,25
330 DATA &H06,&H50      :' L1:  LD   B,80
340 DATA &H77           :' L2:  LD   (HL),A
```

```
350 DATA &H23          :'      INC  HL
360 DATA &H10,&HFC     :'      DJNZ L2
370 DATA &H19          :'      ADD  HL,DE
380 DATA &H0D          :'      DEC  C
390 DATA &H20,&HF6     :'      JR   NZ,L1
400 DATA &HC9          :'      RET
Ok
```

• 画面いっぱいに"♥"を表示するプログラムです.

• 1回目は，140行から160行のBASICプログラムで表示します．2回目は，260行以下のDATA文で定義されている機械語で実行します.

• まず，210行から230行の間で，POKE文を使ってメモリのE000H番地から，機械データを書き込んでいます.

• 180行で，変数VAD%にテキストVRAMの開始番地を代入し，変数ACに機械語サブルーチンの開始番地E000Hを代入しています.

• 190行でCALL文により実行しています.

C

参照：USR

CDBL

コンバート・ダブル：convert to double

機能 整数値，単精度実数値を倍精度実数値に変換した値を返す

書式 CDBL(〈数式〉)

解説
- 〈数式〉の値を倍精度実数値に変換します．ただし，型変換が行われるだけで有効桁数の変化はありません．
- 結果の値の精度は，変換する前の型と同じ(整数型なら整数部のみ，単精度実数型なら有効数字7桁)になります．

例

CDBL(−3) 返される値 −3

CDBL(1.23) 返される値 1.230000019073486

プログラム例

```
100 ' CDBL ｻﾝﾌﾟﾙ
110 A=1.23456:GOSUB 150
120 A=123456!:GOSUB 150
130 A=1.23456E+08:GOSUB 150
140 END
150 A#=CDBL(A)
160 PRINT A,A#
170 RETURN
Ok
run
 1.23456        1.234559893608093
 123456         123456
 1.23456E+08    123456000
Ok
```

- 3つの単精度実数を倍精度実数に変換して出力します．
- 150行目からのサブルーチンで，変数Aに与えられている単精度実数を倍精度実数に変換して表示します．

参照：CINT，CSNG

CHAIN

ディスク

チェイン：chain

機能 プログラムを連結し，実行する

書式 CHAIN〔MERGE〕〈ファイルディスクリプタ〉〔,〈行番号〉〕〔,ALL〕〔,DELETE〈範囲〉〕

解説
- 〈ファイルディスクリプタ〉で指定されたプログラムをロードし，実行します．
- メモリ上にあるプログラム中で，すでに開かれているファイルはその状態を保持します．
- 〈行番号〉は呼び出されたプログラムの実行開始行を指定します．省略した場合は，プログラムの最初から実行します．ここで指定する行番号は，RENUMコマンドを実行して書き換えることはできません．また，ラベル名も使うことはできません．
- ALLオプションを付けると，呼び出されたプログラムにすべての変数，配列を引き渡します．省略した場合は，いっさい引き渡されません．必要な変数，配列だけを引き渡す場合はCOMMON文を使います．
- MERGEオプションは，現在メモリ上にあるプログラムと〈ファイルディスクリプタ〉で指定したプログラムファイルをメモリ上で結合し，結果のプログラムを指定した〈行番号〉から実行します．
- 指定したプログラムファイルは，前もってアスキーセーブしたものでなければいけません．
- メモリ上のプログラムとファイル中のプログラムに同一行番号があった場合，メモリ上の行はファイルの中の行に置き換わります．
- DELETEオプションはMERGEオプションを指定したときのみ意味を持ち，プログラムを結合する前にメモリ上のプログラムの指定した範囲を削除します．ここで指定する行番号の範囲は，RENUMコマンドを実行することによって書き換えられます．また，行番号の代わりにラベル名を使うことができます．

例

```
CHAIN MERGE "1:test",500,ALL,DELETE 500-600
```

- ドライブ1に入っている"test"というファイル名のプログラムファイルをメモリ上にあるプログラムに結合し，500行から実行を開始します．この場合変数，配列はすべて引き渡されます．また，プログラムが結合される前に，メモリ上のプログラムの500行から600行までの削除を行います．

```
100 ' CHAIN ｻﾝﾌﾟﾙ
110 PRINT "< 連結する側のプログラム結果 >"
120 DIM C%(10)
130 A$="N88-BASIC ":B=1.23
140 FOR I=0 TO 10
150   C%(I)=I
160 NEXT I
170 PRINT A$:PRINT B
180 FOR I=0 TO 10
190   PRINT C%(I);
200 NEXT I
210 PRINT:PRINT
220 CHAIN MERGE "2:mergeprog",500,ALL
Ok
run⏎
< 連結する側のプログラム結果 >
N88-BASIC
 1.23
 0  1  2  3  4  5  6  7  8  9  10

< 連結される側のプログラム結果 >
N88-BASIC N88-BASIC
 2.46
 0  2  4  6  8  10  12  14  16  18  20
Ok
```

• "mergeprog"ファイル

```
500 PRINT "< 連結される側のプログラム結果 >"
510 PRINT A$+A$
520 PRINT B+B
530 FOR I=0 TO 10
540   C%(I)=C%(I)+C%(I):PRINT C%(I);
550 NEXT I:PRINT
560 END
Ok
```

• 100行から220行までのプログラムに，ドライブ2に入っているフロッピーディスクの"mergeprog"というファイルを結合して，その実行結果を表示します．

• 130行から160行で，それぞれの変数に数値や文字列を代入し，170行から200行で表示します．

• 220行で，ドライブ2に入っているフロッピーディスクの"mergeprog"ファイルをメモリ上にあるプログラムと結合させ，500行から実行を開始します．このとき，変数はすべて引き渡されます．

• 500行以降の"mergeprog"ファイルでは，引き渡された文字変数A$はそのまま表示し，変数Bおよび配列変数C%は加算を行ってから表示しています．

注意：• MERGEオプションとALLオプションを両方とも指定しない場合は，DEFINT, DEFSNG, DEFDBL, DEFSTR, DEF FN, OPTION BASE, ON ERROR GOTO等の宣言文

は無効になります．また，ALLオプションだけを指定しても，DEF FN, OPTION BASE, ON ERROR GOTO等は無効になります．

参照：COMMON, MERGE, SAVE

CHR$

キャラクタ・ダラー：character $

機能 キャラクタコード，シフトJISコードに対応する文字を返す

書式 CHR$(〈数式〉)

解説
- 〈数式〉の値をキャラクタコードとし，対応する文字またはコントロールコードを返します．
- 〈数式〉は0～255の値とします．もしこの範囲内にない場合は，"Illegal function call" エラーとなります．
- N_{88}-日本語BASICの日本語モードでは，上記に加えて〈数式〉の値をシフトJISコードに対応する全角文字を返します．
- グラフィックシンボルモードでは，N_{88}-BASICと同じように動作します．
- 〈数式〉は以下の範囲とし，もしこの範囲にない場合は，"Illegal function call" エラーとなります．

  00H～FFH(半角文字)

  81yyH～9FyyH, E0yyH～FCyyH (全角文字)

  (yy→00H～FFH)

例

CHR$(&H41) 返される値 "A"

CHR$(12) 返される値 画面消去コード

CHR$(&H889F) 返される値 "亜"

プログラム例

```
100 ' CHR$ ｻﾝﾌﾟﾙ
110 INPUT "number(0-255)";A
120 IF A<0 OR 255<A THEN END
130 PRINT "ｷｬﾗｸﾀ ｺｰﾄﾞ";A;" の文字は ";CHR$(A);" です。"
140 GOTO 110
Ok
run
number(0-255)? 57
ｷｬﾗｸﾀ ｺｰﾄﾞ 57  の文字は 9 です。
number(0-255)? 65
ｷｬﾗｸﾀ ｺｰﾄﾞ 65  の文字は A です。
number(0-255)? 34
ｷｬﾗｸﾀ ｺｰﾄﾞ 34  の文字は " です。
number(0-255)? 42
ｷｬﾗｸﾀ ｺｰﾄﾞ 42  の文字は * です。
number(0-255)? -1
Ok
```

- キャラクタコードを読み込んで，それに対応する文字を出力します．
- 120行で 0 から255以外の値を読み込んだ場合，プログラムを終了します．
- 130行でCHR$関数を使って文字を出力しています．

参照： ASC，資料 4 「キャラクタコード」，資料 8 「日本語コード」

CINT

コンバート・インテジャー：convert to integer

機　能　単精度実数値，倍精度実数値を整数値に変換する

書　式　CINT(〈数式〉)

解　説

- 〈数式〉の値の小数部分を切り捨てて整数型に変換した値を返します．
- 結果の値が－32768～32767の範囲にないときには，"Overflow" エラーが起こります．
- この関数は，INT関数のようにただ数値を整数化するだけでなく，型の変換も行います．

例

CINT(1.2345)　返される値 ⇒ 1

CINT(－3.456)　返される値 ⇒ －3

CINT(56789)　返される値 ⇒ "Overflow" エラー

プログラム例

```
100 ' CINT ｻﾝﾌﾟﾙ
110 INPUT "A,D":A,D%
120 GOSUB 140
130 END
140 AD=CINT(A*10^D%+.5)/10^D%
150 PRINT A,AD
160 RETURN
Ok
run⏎
A,D? 1.2345,3⏎
 1.2345         1.235
Ok
```

- 小数点以下を任意の桁数で四捨五入します．
- 変数Aは値，D%は桁数を表します．
- 140行では，まずAに$10^{D\%}$をかけて，必要な桁数を整数部に持っていきます．それに0.5を加えCINT関数によって小数点以下を切り捨てることにより四捨五入します．最後に$10^{D\%}$で割って小数部であった部分をもとに戻しています．

参照：CSNG, CDBL, INT

CIRCLE

サークル：circle

機能 円(楕円)を描く

書式 CIRCLE {(Wx , Wy) | STEP(x , y)} ,〈半径〉〔,〈パレット番号〉〕〔,〈開始角度〉〕〔,〈終了角度〉〕〔,〈比率〉〕

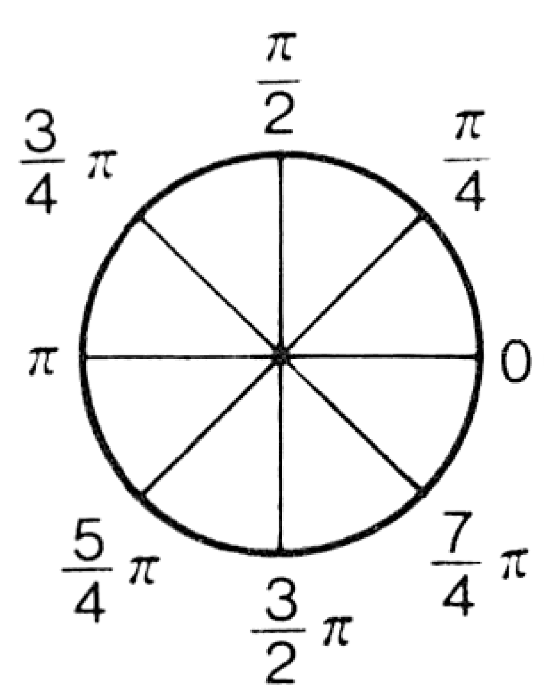
解説

• ワールド座標(Wx, Wy)を中心とし,〈半径〉で指定される大きさの円を描きます.

•〈パレット番号〉が指定されると，指定されたパレットの色で円を描きます．省略された場合には，COLOR文で指定されている〈フォアグラウンドカラー〉で円を描きます.

•〈開始角度〉,〈終了角度〉が指定されると，指定された角度の範囲内に円弧を描きます．角度の指定の単位はラジアン(π/180×角度)で，省略値はそれぞれ0と2πです．〈開始角度〉,〈終了角度〉は−2π〜2πの範囲で指定します．範囲を超えた場合"Illegal function call"エラーとなります.

•〈開始角度〉,〈終了角度〉が負であった場合には，その絶対値をとり正にした角度が用いられますが，そのとき中心から半径が描かれますので扇形を描くことができます.



π＝3.14159

•〈比率〉は，(垂直方向の半径)/(水平方向の半径)で指定します．省略された場合には，640×200ドットのモードでは0.5，640×400ドットのモードでは1が用いられます.

•〈比率〉が1以下の場合には,〈半径〉は水平方向の半径を意味します．また1以上の場合には,〈半径〉は垂直方向の半径を意味します.

• CIRCLE文を実行すると，LPは円の中心座標(Wx, Wy)に設定されます.

例

```
CIRCLE(100,100),50
```

• ワールド座標(100, 100)を中心に，半径50の円を描きます.

プログラム例

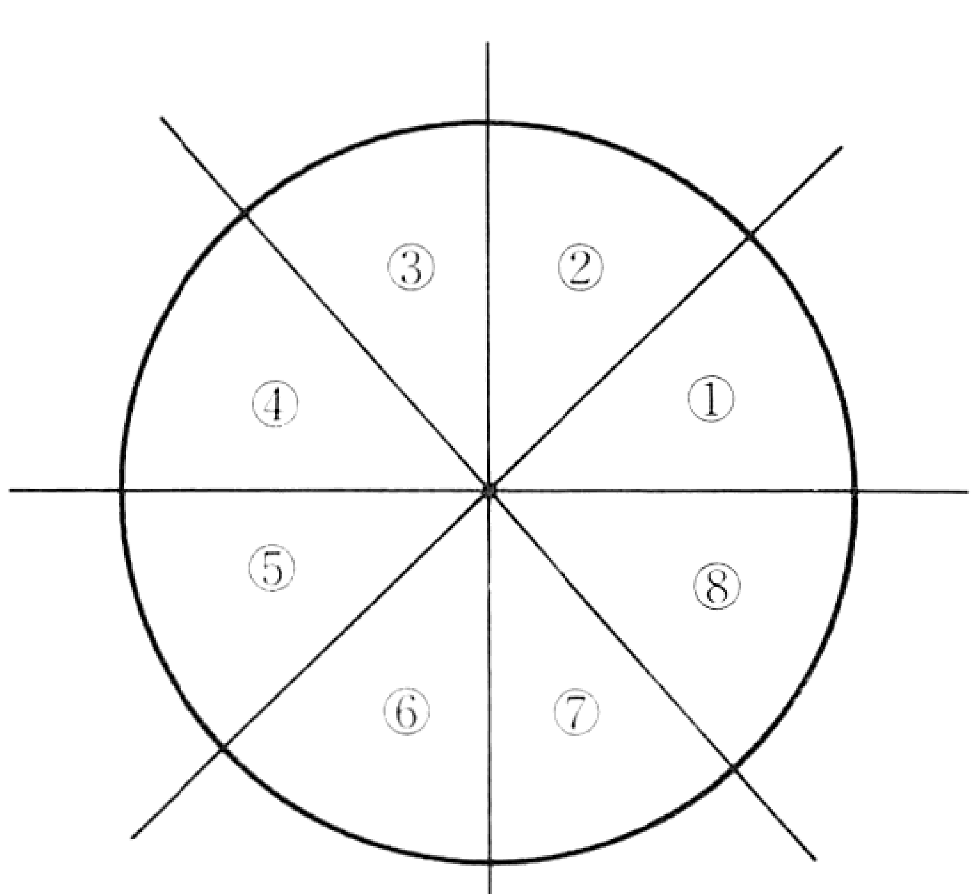
```
100 ' CIRCLE ｻﾝﾌﾟﾙ
110 SCREEN 0,0:CLS 3
120 DATA 25,5,40,13,17
130 FOR I=1 TO 5
140   READ DAT:EN=ST+DAT/100*3.1415*2
150   CIRCLE(320,100),150,I,-ST,-EN
160   ST=EN
170 NEXT I
180 END
Ok
```

- 円グラフを描きます．
- 変数ST，ENは，それぞれ開始角度，終了角度を表します．
- 120行のデータを比率として扇形を続けて描くことによって，円グラフを描きます．

注意：• N_{88}-BASICでは円の始点，終点の座標計算で三角関数を用いていないため，三角関数で求めた座標と異なる描画をします．

そこで，角度指定を行う際に次のような変換を行うことにより，三角関数によって算出した座標に合わせることができます．

θの補正値をwとし，下記①〜⑧の式または式で求めた値(w)をCIRCLE文の開始角度/終了角度として使用します．θが③の領域にあるときは，③式を使用します．なお，θおよびwの単位はラジアンです．



① w＝1.11072＊SIN(θ)

② w＝1.5708－1.11072＊SIN(1.5708－θ)

③ w＝1.5708＋1.11072＊SIN(θ－1.5708)

④ w＝3.14159－1.11072＊SIN(3.14159－θ)

⑤ w＝3.14159－1.11072＊SIN(θ－3.14159)

⑥ w＝4.71239－1.11072＊SIN(4.71238－θ)

⑦ w＝4.71239－1.11072＊SIN(θ－4.71238)

⑧ w＝6.28318－1.11072＊SIN(6.28318－θ)

例：下記は，開始角度＝０，終了角度＝1.3(②の領域にある)の例です．

- グラフィック座標

  CIRCLE(300，100)，150，4，0，1.3

- 補正値によるグラフィック座標

  CIRCLE(300，100)，150，4，0，1.5708－1.11072＊SIN(1.5708－1.3)

参照：COLOR

CLEAR

クリア：clear

機能　変数の初期化およびメモリ領域の設定を行う

書式　CLEAR〔〈ストリング領域の大きさ〉〔,〈メモリの上限〉〕〕〔,〈スタックの大きさ〉〕

解説

- すべての数値変数を 0 に，文字変数を""(ヌルストリング)に初期化します．また，配列宣言も無効となります．
- 〈ストリング領域の大きさ〉は意味を持たないパラメータになっており，指定されても何の影響も与えません．必要なストリング領域を自動的に確保します．
- 〈メモリの上限〉はBASICが使用するメモリの上限番地を指定します．その番地以後に置かれたデータや機械語プログラムは，BASICによって破壊されることはありません．
- 〈スタックの大きさ〉は，BASICがFOR, GOSUB, PAINT等に使用するスタック領域の大きさをバイト数で指定します．リセット時の初期化された状態では，512に設定されています．
- CLEAR文が実行される以前に，DEF文(DEF FN文，DEF USR文，DEF INT文)によって定義あるいは指定された情報は，CLEAR文の実行によってすべて無効となります．

例

CLEAR

- 変数を初期化します．

CLEAR , &HCFFF

- 変数を初期化し，BASICで使用するメモリの上限をCFFFH番地とします．

プログラム例

```
100 ' CLEAR ｻﾝﾌﾟﾙ
110 A%=213:B=74.7543:C#=76346.43263654303#
120 GOSUB 160
130 CLEAR
140 GOSUB 160
150 END
160 PRINT "A%=";A%,"B!=";B!,"C#=";C#
170 PRINT
180 RETURN
Ok
run⏎
A%= 213         B!= 74.7543     C#= 76346.43263654303

A%= 0           B!= 0           C#= 0

Ok
```

- CLEAR文を実行する前と実行した後の変数の値を出力します．
- 130行で数値変数を 0 に初期化しています．

注意：• BASICで使用するスタックは，〈メモリの上限〉を起点として番地の小さい方（0に近い方）へ順に取られていきます．したがって，モニタやPOKE文で〈メモリの上限〉より0に近い方の番地の内容を書き換えますと，プログラムが暴走したり，データが壊れることがあります．

また，CLEAR文が先に実行されていた場合，NEW CMD文の動作は保障されません．

参照：FRE，**プログラマーズガイド**　第20章「機械語プログラムを呼ぶ」

CLOSE

ディスク

クローズ：close

機能 ファイルを閉じる

書式 CLOSE〔〔#〕〈ファイル番号〉〔,〔#〕〈ファイル番号〉〕...〕

解説 •〈ファイル番号〉に対応するファイルを閉じます．以後，CLOSE文によって指定された〈ファイル番号〉は，異なるファイルを開くために再び利用することができます．また，閉じられたファイルは，同じあるいは異なったファイル番号によって再び開くことができます．

•CLOSE文では，〈ファイル番号〉を複数指定することにより一度に複数のファイルを閉じることができます．〈ファイル番号〉が省略された場合には，そのとき開いているファイルをすべて閉じます．

•閉じているファイルに対して，データの入出力を行うことはできません．

•ファイルが出力用にオープンされていた場合には，必ずCLOSE文を実行しなければなりません．

•Rオプション指定のないRUNコマンド，END文，NEWコマンドを実行すると自動的にすべてのファイルを閉じます．STOP文または，[停止]でプログラムの実行を中止した場合，ファイルを閉じる作業は行いませんので注意してください．

例 CLOSE

•オープンされているすべてのファイルを閉じます．

CLOSE #2

•ファイル番号2のファイルを閉じます．

プログラム例

ディスク

```
100 ' CLOSE ｻﾝﾌﾟﾙ
110 OPEN "2:data4" FOR OUTPUT AS #1
120  PRINT #1,"file data 1"
130 CLOSE #1
140 OPEN "2:data5" FOR OUTPUT AS #1
150  PRINT #1,"file data 2"
160 CLOSE #1
170 END
Ok
```

•ドライブ2に入っているフロッピーディスクに"data 4"，"data 5"というファイルを作ります．

•"file data 1"という文字データを"data 4"に，"file data 2"という文字データを"data 5"にそれぞれ書き込みます．

注意：• ファイルがオープンされたままの状態でフロッピーディスクを取り出すと，そのフロッピーディスクの内容を壊すことがあります．したがって，フロッピーディスクを取り出す前にCLOSEを実行してください．

参照：OPEN, END, NEW

CLS

シー・エル・エス：clear screen

機能 画面を消去する

書式 CLS〔〈機能〉〕

解説 •〈機能〉は1，2，3の値をとり，それぞれ次のように働きます．
省略された場合には，1が選択されます．

1：テキスト画面のスクロールウィンドウ内をクリアします．テキスト表示モードがCONSOLE文，COLOR文により白黒リバースモードになっている場合には，画面は白くなります．

N88-日本語BASICの日本語モードでは，スクロールウィンドウ内をテキスト，グラフィックスの区別なくクリアします．

2：グラフィック画面のビューポート内のみをクリアします．グラフィック表示がSCREEN文によりカラーモードに設定されている場合には，COLOR文により指定されたバックグラウンドカラーで，ビューポート内をクリアします．

N88-日本語BASICの日本語モードでは，ビューポート内をテキスト，グラフィックスの区別なくクリアします．

3：テキスト画面，グラフィック画面の両方をクリアします．
テキスト画面がクリアされるときは，CONSOLE文によって指定されたスクロールウィンドウ内のみをクリアします．またグラフィック画面をクリアした場合には，LPはビューポートの左上の頂点に移動します．

N88-日本語BASICの日本語モードでは，スクロールウィンドウ内および，ビューポート内が，テキスト，グラフィックスの区別なくクリアされます．

プログラム例

```
100 ' CLS ｻﾝﾌﾟﾙ
110 SCREEN 0,0
120 GOSUB 180:BEEP:CLS 1
130 GOSUB 180:BEEP:CLS 2
140 GOSUB 180:BEEP:CLS 3
150 GOSUB 180:VIEW(200,50)-(400,150)
160 BEEP:CLS 3
170 END
180 FOR I=1 TO 20
190   FOR J=1 TO 50:PRINT "*";:NEXT J
200   LINE(0,0)-(I*30,199),7
210 NEXT I
220 RETURN
Ok
```

・テキスト画面に"＊"を表示し，グラフィック画面に線を引いて，CLS 1, CLS 2, CLS 3を実行します．

・150行でビューポートを設定しているため，160行のCLS 3では，グラフィック画面はビューポートの中だけ消去されます．

参照：SCREEN, COLOR, VIEW

COLOR

カラー：color

機能 テキスト画面とグラフィック画面の色の指定を行う

書式 COLOR〔〈ファンクションコード〉〕〔,〈バックグラウンドカラー〉〕〔,〈ボーダカラー〉〕〔,〈フォアグラウンドカラー〉〕

解説 •〈ファンクションコード〉は，テキスト画面に表示する文字の色などをカラーコードを使って設定します．この機能は，テキスト画面がカラーモードのときと白黒モードのときとで働きが異なります．

白黒モードの場合(CONSOLE , , , 0)

0 ―ノーマル(通常の表示)
1 ―シークレット(文字を表示しない)
2 ―ブリンク(点滅する)
3 ―シークレット(1と同じ)
4 ―リバース(反転する)
5 ―リバースシークレット(反転して文字は表示されない)
6 ―リバースブリンク(反転して点滅する)
7 ―リバースブリンクシークレット(5と同じ)

> N_{88}-日本語BASICの日本語モードでは,〈ファンクションコード〉は以下のように働きます．
>
> 0 ――― ノーマル(通常の表示)
> 1, 2, 3 ― 0と同じです．
> 4 ――― リバース(反転する)
> 5, 6, 7 ― 4と同じです．

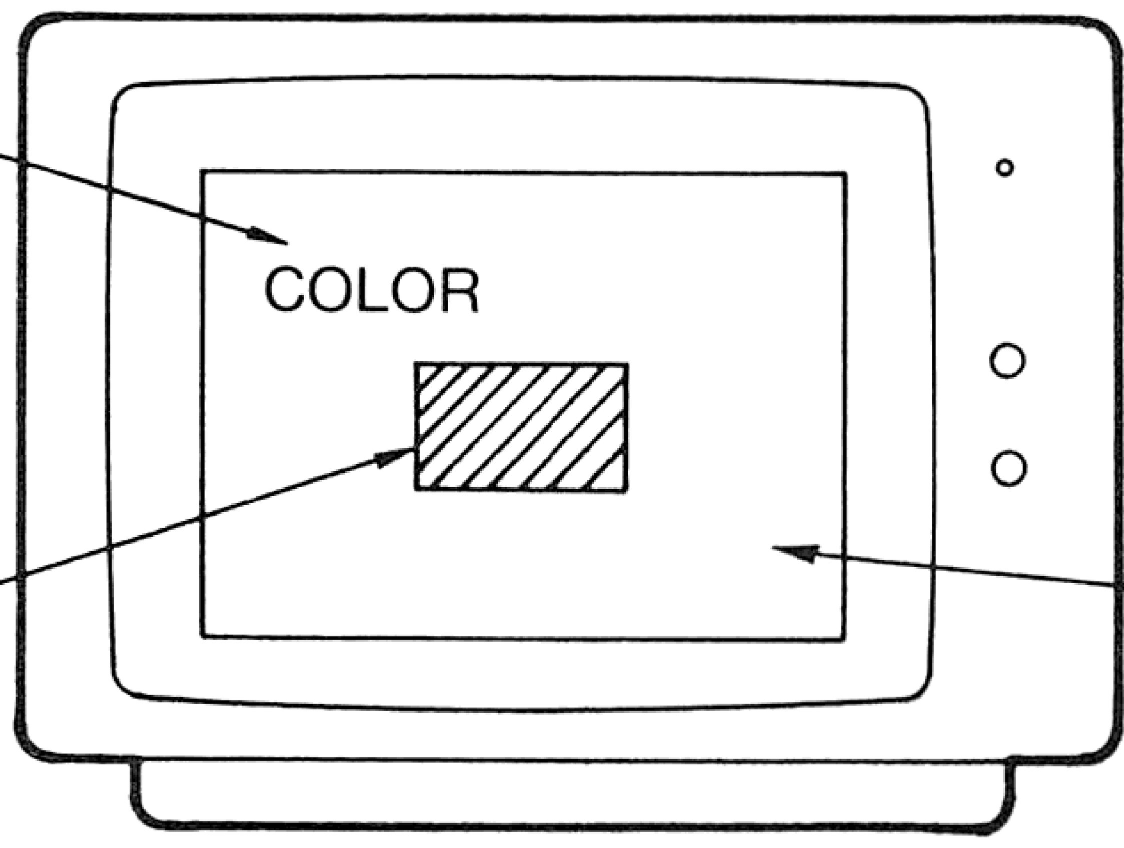
カラーモードの場合(CONSOLE , , , 1)

0 ―黒,	1 ―青,	2 ―赤,	3 ―紫
4 ―緑,	5 ―水色,	6 ―黄色,	7 ―白

•〈バックグラウンドカラー〉とは，グラフィック画面の地の色のことです．グラフィック画面がカラーモードのときは，CLS文によってグラフィック画面を消去するときこの色で塗りつぶされ，パレット番号で指定します．

•〈ボーダカラー〉は，PC-8801$_{MK}$II以降の機種では意味を持ちません．PC-8801のN_{88}-BASICとのコンパチビリティを保つためにあるものですから，普通は省略して使ってください．

•〈フォアグラウンドカラー〉とは，グラフィック画面に点や線を描くときに使われるパレット番号のことです．種々のグラフィックステートメントでパレット番号を指定しなかった場合，この〈フォアグラウンドカラー〉が用いられます．



例　COLOR ,0,,7

•グラフィック画面のフォアグラウンドカラーを7(パレット番号7)，バックグラウンドカラーを0(パレット番号0)に設定します．

プログラム例

```
100 ' COLOR ｻﾝﾌﾟﾙ
110 SCREEN 0,0:CONSOLE ,,,1
120 COLOR 7,1,,7:CLS 3
130 GOSUB 180
140 FOR I=1 TO 7
150   COLOR I,,,I:GOSUB 180
160 NEXT I
170 END
180 PRINT "COLOR"
190 LINE(100,50)-STEP(200,100),,BF
200 FOR J=0 TO 500:NEXT J
210 RETURN
Ok
```

•黒以外の7色で，四角形を描きます．

•120行でテキスト画面の文字を白，バックグラウンドカラーを青，フォアグラウンドカラーを白に設定しています．次のCLS 3が実行されると，グラフィック画面が青になります．

•180行の"COLOR"という文字は，直前に実行されたCOLOR文の〈ファンクションコード〉で指定された色で出力されます．

•190行のLINE文では，直前に実行されたCOLOR文の〈フォアグラウンドカラー〉で指定された色で四角形を描きます．

参照：CLS, CONSOLE, SCREEN, CMD PAL

COLOR=

カラー・イコール：color ＝

機能 カラーパレットを変更する

書式 COLOR＝(〈パレット番号〉,〈カラーコード〉)

解説 • グラフィック画面への色の指定には，カラーパレットを用います．この命令は，どのパレットにどの色を対応させるかを決めるものです．

• 〈パレット番号〉とは，0から7までの8つのカラーパレットにつけられた固有の番号で，それぞれのパレットにどの色を持ってくるかは〈カラーコード〉によって指定します．（詳しくは**プログラマーズガイド** 第10章 「カラー表示」参照．）

• 〈パレット番号〉と〈カラーコード〉の関係は次のとおりです．

〈パレット番号〉		〈カラーコード〉
0		0 (黒)
1		1 (青)
2	どのカラーパレットにどのカラーコードを対応させるかを任意に決めることができる	2 (赤)
3		3 (紫)
4		4 (緑)
5		5 (水色)
6		6 (黄色)
7		7 (白)

• システム起動時には，カラーパレットはそのパレット番号と同じカラーコードが割り付けられています．したがって，もしカラーパレットを全く変更しなければ，カラーパレットとカラーコードは同じものとして使ってかまいません．

• カラーパレットの変更後は，グラフィック画面に表示されている色，およびこれから使うすべてのグラフィック命令(PSET文，LINE文など)におけるカラー表示は，そのパレット番号に割り付けられているカラーコードに設定されます．

• SCREEN 0ではパレットは，グラフィック画面に対してのみ有効です．したがって，テキスト画面のカラーはパレットの変更に影響されません．

例 COLOR＝(1,4)

• カラーパレット1に，カラーコード4(緑)を割り付けます．

プログラム例

```
100 ' COLOR= ｻﾝﾌﾟﾙ
110 SCREEN 0,0:CLS 3
120 C=1
130 FOR I=0 TO 50
140   CIRCLE(320,I+60),I*7,C,,,.3
150   C=C+1:IF C>7 THEN C=1
160 NEXT
170 FOR I=1 TO 7
180   COLOR=(I,0)
190 NEXT
200 FOR N=1 TO 20
210   FOR I=1 TO 7
220     COLOR=(I,1)
230     GOSUB *TIMER
240     COLOR=(I,0)
250   NEXT
260 NEXT
270 FOR I=0 TO 7
280   COLOR=(I,I)
290 NEXT
300 END
310 *TIMER
320  FOR T=0 TO 100:NEXT
330 RETURN
Ok
```

• グラフィック画面に描かれた図形の色を次々に変えることで，アニメーションのように見せます．

• 130行から160行で，パレット番号を1から7に次々に変えながら半径の異なる楕円を描いています．

• 170行から190行でCOLOR＝文を使っていったんすべての色を黒にしたのち，200行から260行でパレット番号1から7を順にカラーコードの1と0に変えることを繰り返し，水の波紋のように見せています．

• 終了後，270行から290行でカラーパレットを初期化しています．

• 320行のFOR～NEXT文のループ回数を変えると，波紋の広がる速さを調節することができます．

注意： • N_{88}-BASIC V2において，テキスト画面がカラーモード並びにグラフィック画面が白黒モードの場合のみ，COLOR＝文でテキスト画面のカラーを変更することができます．

参照：プログラマーズガイド　第10章「カラー表示」，COLOR, CMD PAL

COLOR @

カラー・アットマーク：color @

機　能　テキスト画面に書かれた文字に色や機能を設定する

書　式　COLOR@(X_1, Y_1)−(X_2, Y_2),〈ファンクションコード〉

解　説　• テキスト画面のキャラクタ座標の2点(X_1,Y_1)，(X_2,Y_2)を対角とする四角形の領域に書かれている文字に色を付けたり，ブリンクなどの機能を設定したりします．

• 指定する座標は，キャラクタ座標で次の範囲となります．

X_1, X_2 ＝0〜(〈桁数〉−1)

Y_1, Y_2 ＝0〜(〈行数〉−1)

〈桁数〉，〈行数〉はWIDTH文で指定された値です．

• 〈ファンクションコード〉はCONSOLE文によって，設定されているモードによって持つ意味が異なります．この機能および指定の仕方はCOLOR文の〈ファンクションコード〉の場合と全く同様ですから，そちらを参照してください．

• もし〈ファンクションコード〉が省略された場合は，7を値として採用します．

例　COLOR@(0,0)−(9,9),4　(カラーモードの場合)

• テキスト画面の左上(10桁×10行)を緑に設定します．

プログラム例

• このプログラムは，N88-日本語BASICではグラフィックシンボルモード(SCREEN文の第1パラメータを0，1，2のいずれかに設定する)で実行してください．

```
100 ' COLOR@ ｻﾝﾌﾟﾙ
110 CONSOLE 0,25,0,1
120 WIDTH 80,25:CLS
130 FOR I=0 TO 15
140   PRINT "COLOR@ test"
150 NEXT I
160 FOR I=0 TO 7
170   COLOR@(0,I*2)-(10,I*2+1),I
180 NEXT I
190 END
Ok
```

• "COLOR@ test"という文字を8色に分けて出力します．

• 130行から150行で，"COLOR@ test"という文字を縦に16行分表示します．

• 160行から180行で，テキスト画面を2行ごとに色を指定します．

注意：・この命令によって設定された領域の上に新しく文字を書いた場合，書かれた文字はこの影響を受けません．

・N_{88}-日本語BASICの日本語モードでは(SCREEN文の第1パラメータが3または4に設定されている状態では)，COLOR@文で文字に色や機能を設定することはできません．実行しようとすると"Illegal function call"エラーになります．

参照：CONSOLE, COLOR

COMMON

ディスク

コモン：common

機能 CHAIN文が実行されたとき変数を引き渡す

書式 COMMON 〈変数名〉〔,〈変数名〉...〕

解説

• COMMON文は，必ずCHAIN文とペアにして使います．

• CHAIN文が実行され，メモリ上のプログラムと外部からのプログラムがチェインされたとき，〈変数名〉で指定された変数を引き渡します．

• COMMON文は非実行文ですから，プログラムのどこに置いてもかまいませんが，CHAIN文の前にある必要があります．

•〈変数名〉は1行の許す範囲で何個でも並べることができますが，1つのプログラム中のCOMMON文に同じ変数名があってはいけません．

• プログラム中のすべての変数を引き渡す場合は，CHAIN文のALLオプションを使う方が便利です．

例 COMMON A , B , XY ( ) , NA$

• 数値変数A，数値変数B，配列変数XY，文字変数NA$を引き渡します．

プログラム例

ディスク

```
100 ' COMMON ｻﾝﾌﾟﾙ
120 COMMON A$,A%,B,C#
130 A$="COMPUTER"
140 A%=34
150 B=3.45
160 C#=1.23456789#
170 CHAIN "2:common2"
180 END
Ok
run⏎
< 連結される側のプログラム結果 >
COMPUTER
A = 68         B = 11.9025    C = 1
Ok
```

• "common2" ファイル

```
500 PRINT "< 連結される側のプログラム結果 >"
510 A%=A%+A%:B=B*B:C#=C#/C#
520 PRINT A$
530 PRINT "A =";A%,"B =";B,"C =";C#
Ok
```

・変数A$, A%, B, C#を引き渡し，画面に表示します.

・170行で"common2" ファイルを呼び出しています.

参照：CHAIN

COM ON/OFF/STOP

コム・オン/オフ/ストップ：communication on/off/stop

機能 通信回線からの割り込み許可，禁止，停止を設定する

書式
1) COM ON
2) COM OFF
3) COM STOP

解説 • コミュニケーションポート(RS-232C)に外部からの通信が入ったことによる割り込みを許可(ON)するか，禁止(OFF)するか，停止(STOP)するかを設定します．

1) 割り込みを許可します．この命令実行後は，コミュニケーションポートに通信が入るごとに割り込みが発生し，ON COM GOSUB文で定義されている処理ルーチンに分岐します．

2) 割り込みを禁止します．この命令実行後はコミュニケーションポートに通信が入っても割り込みは発生せず，処理ルーチンへの分岐は起こりません．

3) 割り込みを停止します．この命令実行後は，通信が入ってもそのことを覚えているだけで処理ルーチンへの分岐は起こりません．しかし，その後COM ONによって割り込みが許可されると，先ほどの通信によって処理ルーチンに分岐します．

例 COM ON

• 通信が入ったことによる割り込みを許可します．

プログラム例

```
100 ' COM ON/OFF/STOP ｻﾝﾌﾟﾙ
110 CLS
120 OPEN "COM1:N81" FOR INPUT AS #1
130 ON COM GOSUB *COMMUNICATE
140 COM STOP
150 IF RIGHT$(TIME$,1)="0" THEN COM ON
160 LOCATE 10,10:PRINT TIME$
170 GOTO 140
180 *COMMUNICATE
190  COM OFF
200  PRINT STRING$(80," ");CHR$(30);
210  A$=INPUT$(1,#1)
220  PRINT A$;
230  IF NOT(EOF(1)) THEN 200 ELSE RETURN
Ok
```

• コミュニケーションポートからの入力による割り込みにより，そのデータを画面に表示するプログラムです．

• 160行で，現在時刻を表示しています．

• 割り込みにより180行に分岐して，そのデータを表示します．割り込みの許可は，秒の下

の桁が0のときだけ行われています.

参照：プログラマーズガイド　第17章「コンピュータコミュニケーション」, ON COM GOSUB, KEY ON/OFF/STOP

CONSOLE

コンソール：console

機能 テキスト画面のモード設定を行う

書式 CONSOLE 〔〈スクロール開始行〉〕〔,〈スクロール行数〉〕〔,〈ファンクションキー表示スイッチ〉〕〔,〈カラー／白黒スイッチ〉〕

解説
- テキスト画面に関するさまざまなモード設定を行います.
- 〈スクロール開始行〉と〈スクロール行数〉を指定することにより，画面上でのスクロールする領域(スクロールウィンドウ)を指定します.
- CLSおよびPRINT CHR$(12)は，このスクロールウィンドウに対して実行されます.
- 〈スクロール開始行〉は，どの行からスクロールを開始するかを指定します.
- 〈ファンクションキー表示スイッチ〉に1を指定すると，画面最下行にプログラマブルファンクションキーに登録されている5個のファンクションキーを表示します. 0を指定した場合は表示されません.
- N_{88}-日本語BASICでは，〈ファンクションキー表示スイッチ〉に2を指定すると10個のファンクションキーを表示します.
- 〈カラー/白黒スイッチ〉を1にするとテキスト画面をカラーモードにし，0を指定した場合には白黒モードとなります.

例 CONSOLE 0, 10, 1, 1
- 最上行から10行をスクロールウィンドウとし，ファンクションキーを表示するカラーモードに設定します.

プログラム例

```
100 ' CONSOLE ｻﾝﾌﾟﾙ
110 CLS:CONSOLE 0,25,0,1
120 FOR I=1 TO 25 STEP 4
130   CONSOLE 0,I
140   GOSUB 170
150 NEXT I
160 END
170 CLS
180 FOR J=1 TO 40
190   PRINT J
200   FOR K=0 TO 10:NEXT K
210 NEXT J
220 RETURN
Ok
```

- スクロール行数を変えて，1から40までの数字を出力します.
- 130行で，スクロール行数を設定しています.

注意：• BASICには，プログラム中でエラーが発生するとその行を表示し，エラー位置を示す機能があります．このとき，その行がスクロールウィンドウ中に入らないと，エラー表示が正常に動作しない場合があります．プログラムのデバッグ中はスクロールウィンドウを大きめにとってください．

• N_{88}-日本語BASICの日本語モード時に第3パラメータを0に指定しても，[全角] を押すことにより，ファンクションキー表示が現れます．再び，[全角] を押して半角文字モードにすると，ファンクション表示が消えます．ただし，半角文字モードで，最下行にカーソルを置いて [全角] を押しても [全角] の働きはしません．

• N_{88}-BASICの場合には，〈ファンクションキー表示スイッチ〉に2を指定してもファンクションキーは5個表示されたままです．

CONT

コント：continue

機能 停止したプログラムの実行を再開する

書式 CONT

解説 •CONT コマンドは，[停止] や，STOP 文，END 文で停止または終了した文の次の文から実行を開始するコマンドです．

•通常は，プログラムの誤りを探すときに，STOP文とともに用います．

•実行を停止した後ダイレクトモードで変数の値を変更したり，表示した後CONTコマンドによって実行を再開することができます．

•実行停止中にプログラム内容の変更を行った場合には，CONT 文による実行再開はできません．

例 CONT

•プログラムの実行を再開します．

実 行 例

```
100 ' CONT ｻﾝﾌﾟﾙ
110 I=10
120 PRINT I,SQR(I)
130 STOP
140 I=I+1
150 GOTO 120
Ok
run↵
 10            3.16228
Break in 130
Ok
cont↵
 11            3.31662
Break in 130
Ok
cont↵
 12            3.4641
Break in 130
Ok
```

•10以上の整数の平方根をSTOP文で止めながら出力します．

•実行例は，130行がなければ10以上の整数とその平方根を次々と出力するものです．

•実行されると，10とその平方根を出力してすぐ停止します．STOP文による停止なので，CONT文を入力することによってSTOP文の次の140行から実行を再開することができます．

参照：STOP, END

COPY

コピー：copy

機能 画面に表示されている情報をプリンタに出力する

書式 COPY〔〈機能〉〕

解説

• テキスト画面，グラフィック画面に表示されている文字や図形をプリンタに出力します．

• 〈機能〉は，1から5までの値をとり，次のように働きます．省略されたときは，1が選択されます．

1：テキスト画面のみをプリンタにコピーします（［コントロール］＋［画面コピー］を押しても同じです）．

2：グラフィック画面のみをプリンタにコピーします（［GRPH］＋［画面コピー］を押しても同じです）．

3：テキスト画面，グラフィック画面の両方をプリンタにコピーします（［画面コピー］を押しても同じです）．

4：グラフィック画面のみをプリンタにコピーします（漢字出力用）．640×200ドットのモードで出力された漢字（グラフィックパターン）は，このモードで出力すると縦方向を2分の1に縮小してコピーします．

5：テキスト画面，グラフィック画面の両方をプリンタにコピーします．4と同じく640×200ドットのモードで出力されたパターンは，縦方向を2分の1に縮小してコピーします．

• N_{88}-日本語BASICでは，〈機能〉1～5は次のように働きます．

1：テキスト画面のみをプリンタにコピーします．このとき，半角文字2文字分のスペースと全角文字1文字分のスペースを同じにするために，全角文字の前後にドットスペースを置いて補正を行います．ただし，ファンクションキーの内容が画面に表示されていてもプリンタにコピーすることはできません．

2，3：N_{88}-BASIC COPY 2と同じです．ただし，テキストもグラフィック画面に書かれていますから，同時に印字されます．

4，5：N_{88}-BASIC COPY 4と同じです．ただし，テキストもグラフィック画面に書かれていますから，同時に印字されます．

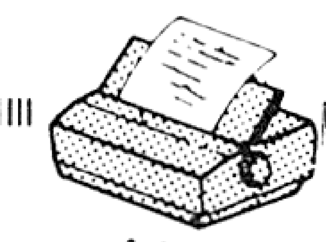
（詳しくは，**プログラマーズガイド** 第11章「カラーコピーユーティリティ」を参照してください．）

例

```
COPY 2
```

• グラフィック画面のみをプリンタに出力します．

プログラム例


プリンタ

```
100 ' COPY ｻﾝﾌﾟﾙ
110 SCREEN 0,0:WIDTH 80,25:CLS 3
120 FOR I=1 TO 20
130   X=RND(1)*75:Y=RND(1)*19
140   LOCATE X,Y:PRINT "COPY";
150   CIRCLE(X*8,Y*10),30
160 NEXT I
170 FOR I=1 TO 5
180   COPY I
190 NEXT I
200 END
Ok
```

• テキスト画面のいろいろな位置に"COPY"という文字を出力し，グラフィック画面のいろいろな位置に円を描いてから，画面をプリンタに出力します．

• 120行から160行で，画面に"COPY"と円を描きます．

• 170行から190行で，COPY 1からCOPY 5まで順番に実行します．

注意：• COPY文が使用可能なプリンタは次のとおりです．これら以外のプリンタに対してのCOPY文の動作は保証されません．

• 縦横の比率がほぼ画面上と同等にコピーできるもの

PC-8024, PC-8027, PC-PR101L/T/TL, PC-PR101F, PC-PR201H/H2, PC-PR201CL/HC, PC-PR201TL, PC-PR201V, PC-PR201F, PC-PR406, PC-PR406M, PC-PR102TL

• 縦長(約4/3倍)になるがコピーできるもの

PC-PR103, PC-PR202K

• COPY文([画面コピー])でグラフィック画面をプリンタへコピーすると，画面とプリンタとの縦横の比が異なる場合があります．

• 次のプリンタ以外では，全角文字と半角文字の桁がそろいません．

PC-PR101, PC-PR101L/T/TL, PC-PR101F, PC-PR201, PC-PR201CL/HC, PC-PR201H/H2, PC-PR201T/TL, PC-PR201F, PC-PR406, PC-PR406M, PC-PR102TL

COS

コサイン：cosine

機能 余弦（コサイン）の値を返す

書式 COS(〈数式〉)

解説
- 〈数式〉の値の余弦（コサイン）を返します.
- 〈数式〉の値の単位は，ラジアン（π/180×角度）です.
- 〈数式〉に倍精度実数が含まれる場合は倍精度の値を返しますが，他の場合には単精度の値を返します.

例

COS(3.14159/180＊30) 返される値 .866026（COS（30°）の値）

COS(3.14159/180＊60) 返される値 .500001（COS（60°）の値）

プログラム例

```
100 ' COS ｻﾝﾌﾟﾙ
110 INPUT "角度 (度) : ";D
120 CS=COS(D/180*3.14159265358979#)
130 PRINT "COS";USING "(###°)";D;:PRINT " = ";CS,
140 PRINT "SEC";USING "(###°)";D;:PRINT " = ";
150 IF ABS(CS)<=.000001 THEN PRINT "定義されません"
                        ELSE PRINT 1/CS
160 END
Ok
run
角度 (度) : ? 45
COS( 45°) =  .707107        SEC( 45°) =  1.41421
Ok
```

- 角度を度で読み込んで，コサイン(COS)とセカント(SEC)の値を出力するプログラムです.
- セカントの値を求めるには

$$\sec\ \theta = 1/\cos\ \theta$$

を用いて計算します.
- cosの値が0.000001以下になると，単精度実数ではEのついた指数形式になるので，150行ではそのような場合はオーバーフローとみなすようにしてあります.

参照：SIN，TAN，ATN

CSNG

コンバート・シングル：convert to single

機能 整数値，倍精度実数値を単精度実数値に変換した値を返す

書式 CSNG(〈数式〉)

解説
• 〈数式〉の値を有効数字6桁の単精度実数値に変換した値を返します．
• 結果の値が −1.70141E+38〜1.70141E+38の範囲にないときは，"Overflow"エラーが起こります．

例
CSNG(123) 返される値 ⇒ 123
CSNG(3.14159) 返される値 ⇒ 3.14159
CSNG(−1E+40) 返される値 ⇒ "Overflow"エラー

プログラム例

```
100 ' CSNG ｻﾝﾌﾟﾙ
110 INPUT"A,D";A#,D%
120 AD#=INT(A#*10^D%+.5)/10^D%
130 PRINT A#,CSNG(AD#)
140 END
Ok
run
A,D? 1.25,1
 1.25          1.3
Ok
run
A,D? 0.1234567,5
 .1234567      .12346
Ok
```

• 小数点以下を任意の桁数で四捨五入します．
• 120行では，CINT関数のプログラム例と同じことを実行していますが，ここでは，CINT関数でなくINT関数を使っています．
• 130行で変数AD#をそのまま出力すると，誤差を含んだ結果が出てくるので，CSNG関数を使って有効桁数はそのままで単精度実数に変換してから出力しています．

参照：CINT，CDBL

CSRLIN

カーソル・ライン：cursor line

機能 カーソルの垂直位置を返す

書式 CSRLIN

解説

• 現在のカーソルの垂直位置を行単位で返します．

• 得られる値は25行モードで0～24，20行モードでは0から19となります．なお画面の最上行が0，最下行が24(または19)となっています．

例 CSRLIN

• 現在のカーソルの垂直位置を返します．

プログラム例

• このプログラムは，N88-日本語BASICではグラフィックシンボルモード(SCREEN文の第1パラメータを0，1，2のいずれかにする)で実行してください．

```
100 ' CSRLIN ｻﾝﾌﾟﾙ
110 WIDTH 80,25:CLS
120 F=1:X=39:Y=2
130 GOSUB 150
140 GOTO 130
150 LOCATE X,Y:PRINT " ";
160 IF CSRLIN>=22 OR CSRLIN<=1 THEN F=F*-1
170 Y=Y+F:LOCATE X,Y:PRINT "●";
180 FOR TT=0 TO 20:NEXT
190 RETURN
Ok
```

• カーソルの位置を調べながら"●"を上下に動かします．

• 160行でカーソルの位置を調べ，1行目か22行目のときに動く方向を変えます．

• 変数Fは方向を表し，1のときは下，−1のときは上を表します．こうすることによって，次のY座標はY+Fで求めることができます．

参照：POS

CVI/CVS/CVD

シー・ブイ・アイ：convert to integer
シー・ブイ・エス：convert to single
シー・ブイ・ディ：convert to double

機能 文字列を数値データに変換した値を返す

書式 CVI(〈2文字の文字列〉)

CVS(〈4文字の文字列〉)

CVD(〈8文字の文字列〉)

解説 • フロッピーディスク上のランダムファイルから読み込んだデータは文字型になっているため，CVI, CVS, CVD関数を使って数値データに変換します．

• CVI関数は2文字の文字列(すなわち2バイトのデータ)を整数値に，CVS関数は4文字の文字列を単精度実数値に，CVD関数は8文字の文字列を倍精度実数値にそれぞれ変換します．

• 指定した関数の文字列より長い文字列を代入した場合，代入した文字列の左側から指定されている文字数までが変換されます．たとえば，CVI("ABCD")とした場合，"AB"だけが変換されます．

例 CVI("AB") 返される値 16961

• 2バイトの文字コード"AB"を整数値16961(内部表現＝4241H)に変換します．数値の内部表現については，VARPTR関数を参照してください．

プログラム例

ディスク

```
100 ' CVI/CVS/CVD ｻﾝﾌﾟﾙ
110 OPEN "2:rdata" AS #1
120  FIELD #1,2 AS D1$,4 AS D2$,8 AS D3$
130  GET #1,1
140  A%=CVI(D1$)
150  B=CVS(D2$)
160  C#=CVD(D3$)
170  PRINT "A =";A%
180  PRINT "B =";B
190  PRINT "C =";C#
200 CLOSE
210 END
Ok
run
A = 123
B = 8534.13
C = 3.14159265358979
Ok
```

• ランダムファイルからデータを読み取り，数値に変換して表示します．

• MKI$/MKS$/MKD$のプログラム例を実行することによって作成された"rdata"というファイルの内容を出力します．

・140行から160行で，2バイトの文字列を整数型，4バイトの文字列を単精度実数型，8バイトの文字列を倍精度実数型に変換しています．

参照：MKI$/MKS$/MKD$

DATA

データ：data

機能 READ文で読み込まれる数値，文字定数を格納する

書式 DATA 〈定数〉〔,〈定数〉...〕

解説

• DATA文は，READ文で読み込まれる数値や文字定数を格納します．

• この命令は，REM文などと同じように非実行文(何の動作も行わない命令)で，プログラムのどこにおいてもかまいません．

• 1つのDATA文には，プログラムの1行(255文字)に入るだけのデータをセットすることができ，また1つのプログラム中にいくつものDATA文を置くことができます．

• READ文は，行番号の小さい方から順番にDATA文の中のデータを読み込んでいきます．

• 〈定数〉は整数，単精度実数，倍精度実数，文字定数のいずれかです．ただし，定数式(2＊3など)は許されません．またREAD文で読み込まれる場合，READ文で指定されている変数の型は，対応するデータの定数の型と一致しなければなりません．

• DATA文の中のデータはコンマ(,)で区切られますが，文字定数でその文字列の先頭，または最後がコンマ，ピリオド(.)，意味のある空白を含むときは，その文字定数全体をダブルクォーテーション(")で囲む必要があります．

• RESTORE文によって，READ文で読み込むDATA文を行単位で指定することができます．

例 DATA 8001,"COMPUTER"

• データとして8001という数値データ，"COMPUTER"という文字列データを格納します．これらはREAD文によって読み出されます．

プログラム例

```
100 ' DATA ｻﾝﾌﾟﾙ
110 READ A,A$
120 PRINT A:PRINT A$
130 DATA 10,20
140 END
Ok
run⏎
 10
20
Ok
```

• DATA文に格納されている定数を読み取り，出力します．

• 130行で，10と20というデータを設定します．

• 110行のREAD文では，AとA$という変数に10，20というデータをそれぞれ代入します．10は単精度実数型，20は文字型として読み込まれます．

参照：READ, RESTORE

DATE$

デート・ダラー：date $

機能 内蔵のカレンダ時計の日付を返す

書式 DATE$

解説

• DATE$変数を参照することにより，内蔵カレンダ時計の日付が得られます．

• 日付は8文字の文字列で返され，文字変数に代入したり，PRINT文で内容を表示することができます．

• 日付を設定するには，DATE$変数に次の形式の文字列を代入します．

"YY/MM/DD"

YY ・・年(00～99)
MM ・・月(01～12)
DD ・・日(01～31)
各2文字で指定します．

例

DATE$ 返される値 (現在の日付を返す)

DATE$＝"86/11/03"

• 内蔵カレンダ時計に，1986年11月03日を設定します．

プログラム例

```
100 ' DATE$ ｻﾝﾌﾟﾙ
110 PRINT "きょうは 19";
120 PRINT LEFT$(DATE$,2);" 年 ";
130 PRINT MID$(DATE$,4,2);" 月 ";
140 PRINT RIGHT$(DATE$,2);" 日 です。"
150 END
Ok
run
きょうは 1986 年 11 月 03 日 です。
Ok
```

• 日付を出力します．

• 120行でDATE$変数の左端から2文字を取り出して，年を出力します．

• 130行でDATE$変数の左端から数えて4番目から2文字を取り出して，月を出力します．

• 140行では，DATE$変数の右端から2文字を取り出して，日を出力します．

注意：• 内蔵のカレンダ時計はバッテリーバックアップにより，常に日付を更新しています．

• うるう年の場合にも自動判定して，日付を正しく返します．

参照：TIME$

DEF FN

ディファイン・ファンクション：define function

機能 新しく関数を定義する

書式 DEF FN 〈名前〉〔(〈パラメータリスト〉)〕=〈関数の定義式〉

解説

• 関数の名前は"FN"+〈名前〉で定義します(例：FNPW).

•〈名前〉は変数名として正しい形をしたものでなくてはなりません(40文字まで有効です).

•〈パラメータリスト〉はその関数が呼ばれたときに，〈関数の定義式〉中の同じ変数名に対応します．変数名の間はコンマで区切ります．

•〈パラメータリスト〉に書かれた変数名は，〈関数の定義式〉を評価する際にのみ有効となります．したがって，プログラム中に同じ変数名があっても影響はありません．

•〈関数の定義式〉はその関数の演算内容を記述する式で，1行の範囲に限られます．このユーザ定義関数は数値関数，文字関数，その混在のいずれでもかまいませんが，〈パラメータリスト〉内の変数と，それに渡される変数とが同一の型でなければなりません．

•〈関数の定義式〉中で使われている変数が〈パラメータリスト〉中にない場合は，その変数がその時点で持っている値が使われます．

• DEF FN文は非実行文ですが，これによって定義される関数がプログラム中で呼ばれる前に実行されていなければなりません．

• DEF FN文は，ダイレクトモードで使用することはできません．

例 DEF FNLG(X)=LOG(X)/LOG(10)

• 常用対数を求める関数として"FNLG"を定義します．以後，プログラム中でFNLG(〈数式〉)を使うと，〈数式〉の常用対数値を得ることができます．

プログラム例

```
100 ' DEF FN ｻﾝﾌﾟﾙ
110 DEF FNPW(A,B)=A^B
120 INPUT "x,y";X,Y
130 PRINT "x^y=";FNPW(X,Y)
140 END
Ok
run↵
x,y? 2,8↵
x^y= 256
Ok
```

• 2つの実数x，yを読み込んで，x^yを出力します．

• 110行で引数x，yを持ち，x^yを返す関数FNPWを定義しています．

参照：1.7.2　変数名と型宣言文字

DEFINT/SNG/DBL/STR

ディファイン・イント/シングル/ダブル/ストリング：define integer/single/double/string

機能 変数の型宣言を行う

書式 DEF {INT | SNG | DBL | STR}〈文字の範囲〉〔,〈文字の範囲〉...〕

解説

・〈文字の範囲〉で指定された文字で始まるすべての変数名の型を，次に示す型宣言により定義します．

DEFINT ：変数を整数型に
DEFSNG：変数を単精度型に
DEFDBL：変数を倍精度型に
DEFSTR：変数を文字型に
} 宣言します．

・〈文字の範囲〉は，1文字の英字で指定します．また，ある範囲にわたって指定するときは，1文字の英字と1文字の英字をハイフン(－)でつなぎます．ただし，最初の英字は，後の英字よりもアルファベット順で先のものでなければなりません．

・型宣言文字で指定した場合，これらのDEF宣言よりも優先されます．

・型宣言が行われていない文字で始まる変数は，すべて単精度実数型変数とみなされます．

例 DEFINT I, J

・IとJで始まる変数を整数として宣言します．以後，プログラム中では整数型として扱われます．たとえば，IXはIX%と等しくなります．

プログラム例

```
100 ' DEF INT/SNG/DBL/STR ｻﾝﾌﾟﾙ
110 DEFINT I-M
120 DEFSNG A,B
130 DEFDBL D
140 DEFSTR S
150 I1=1.23:J=65.643
160 AB=1.23:BB=65.643
170 D=3.141592653589792#
180 D%=3.141592654000001#
190 S=" STRING"
200 PRINT "I1 =";I1,,"J =";J
210 PRINT "AB =";AB,,"BB =";BB
220 PRINT "D =";D,"D% =";D%
230 PRINT "S =";S
240 END
Ok
run⏎
```

```
I1 = 1                        J = 66
AB = 1.23                     BB = 65.643
D = 3.141592653589792         D% = 3
S = STRING
Ok
```

• 整数型，単精度実数型，倍精度実数型の数を，それぞれ出力するプログラムです．

• 110行でI, J, K, L, Mで始まる変数を整数型，120行でA, Bで始まる変数を単精度実数型，130行でDで始まる変数を倍精度実数型，140行でSで始まる変数を文字型にそれぞれ定義します．

• 180行では，Dで始まる変数名に倍精度実数値を代入していますが，型宣言文字%で整数型に指定されているので，D%という変数は3という値を持つことになります．

DEF USR

ディファイン・ユーザ：define user

機能 機械語で作られたユーザ関数の実行開始番地を定義する

書式 DEF USR〔〈番号〉〕=〈開始番地〉

解説
- USR関数(ユーザ関数)が呼び出す機械語プログラムの実行開始番地の設定を行います.
- 〈番号〉は0～9までの値で，USR＋〈番号〉が関数名となり，関数名によって機械語で作られたプログラムが呼び出されます.〈番号〉を省略したときは0とみなされます.
- 〈開始番地〉は機械語プログラムの実行開始番地を指定します.

例 DEF USR3 = &HDA00

- USR関数の実行開始番地を，DA00H番地に設定します.

プログラム例

- このプログラムは，N₈₈-BASICで実行してください.

```
100 ' DEF USR ｻﾝﾌﾟﾙ
110 CLEAR ,&HDFFF
120 DEF USR=&HE000:GOSUB 210
130 CONSOLE 0,25,0,1:WIDTH 80,25:CLS
140 FOR I=1 TO 80*24
150   PRINT "♥";
160 NEXT I
170 CLS:FOR I=1 TO 500:NEXT I
180 VAD%=&HF3C8
190 I=USR(VAD%):FOR I=1 TO 500:NEXT I
200 END
210 FOR AD=&HE000 TO &HE015
220   READ DA:POKE AD,DA
230 NEXT AD
240 RETURN
250 '
260 DATA &H7E          :'      LD   A,(HL)
270 DATA &H23          :'      INC  HL
280 DATA &H66          :'      LD   H,(HL)
290 DATA &H6F          :'      LD   L,A
300 DATA &H3E,&HE9     :'      LD   A,"♥"
310 DATA &H11,&H28,&H00 :'     LD   DE,40
320 DATA &H0E,&H19     :'      LD   C,25
330 DATA &H06,&H50     :' L1:  LD   B,80
340 DATA &H77          :' L2:  LD   (HL),A
350 DATA &H23          :'      INC  HL
360 DATA &H10,&HFC     :'      DJNZ L2
370 DATA &H19          :'      ADD  HL,DE
380 DATA &H0D          :'      DEC  C
390 DATA &H20,&HF6     :'      JR   NZ,L1
400 DATA &HC9          :'      RET
Ok
```

- BASICと機械語のプログラムで，テキスト画面全体に"♥"を2回書きます.
- 1回目は140行から160行のBASICプログラムで実行しますが，2回目は260行以下のDATA文で定義されている機械語で実行します.

・120行で，機械語の開始番地がE000H番地であることを定義します.

・180行で変数VAD%にテキストVRAMの開始番地を代入し，190行でそれを引数として機械語ルーチンを呼び出します.

・210行から240行までのサブルーチンで，DATA文に定義されている機械語データをメモリに書き込みます.

参照：USR，**プログラマーズガイド**　第20章　「機械語プログラムを呼ぶ」

DELETE

デリート：delete

機能 プログラムを行単位で削除する

書式 DELETE〔〈始点行番号〉〕〔－〈終点行番号〉〕

解説
- 〈始点行番号〉から〈終点行番号〉までのプログラム行を削除します.
- 〈始点行番号〉だけを指定した場合はその行だけを削除します. この場合, DELETEコマンドを省略して行番号だけを入力したのと同じになります.
- ハイフン(－)と〈終点行番号〉のみを指定した場合, プログラムの先頭から指定行までを削除します.
- 〈始点行番号〉あるいは〈終点行番号〉の代わりに, 現在の行を表すピリオド(.)を使用することができます.
- 〈始点行番号〉と〈終点行番号〉の両方を省略して使うことはできません.

例 DELETE －990

- プログラムの先頭から990行までを削除します.

実行例

```
100 ' DELETE ｻﾝﾌﾟﾙ
110 '
120 PRINT " DELETE コマンドは 指定した行を"
130 PRINT " 削除します"
140 PRINT " DELETE サンプル"
150 PRINT " DELETE コマンドは 指定した行を"
160 PRINT " 削除します"
170 END
Ok
delete 110-130 ⏎
Ok
list⏎
100 ' DELETE ｻﾝﾌﾟﾙ
140 PRINT " DELETE サンプル"
150 PRINT " DELETE コマンドは 指定した行を"
160 PRINT " 削除します"
170 END
Ok
```

- 100行から170行までのプログラムのうち, 110行から130行までをDELETEコマンドを使って削除しています.

DIM

ディメンジョン：dimension

機能　配列変数の要素の大きさを指定する

書式　DIM〈変数名〉(〈添字の最大値〉〔,〈添字の最大値〉...〕)

解説
- 配列を使用する前には，DIM文によって配列宣言を行います．
- DIM文は配列変数の添字の最大値を設定し，同時にメモリ上にその配列の領域を割り当てるものです．
- 添字の最大値が10以下の場合は，配列宣言をする必要はありません．このとき，配列を使用した時点でメモリ上に，添字の最大値を10として領域を割り当てます．
- 〈添字の最大値〉の個数は，その配列変数の次元を示します．また，次元はメモリ容量の許す限り255次元まで使用できます．
- 〈添字の最大値〉より大きい値の添字を持つ配列が用いられた場合は，"Subscript out of range"エラーになります．
- 添字の最小値は，OPTION BASE文によって，0か1に指定します．
- DIM文が実行された時点で，その配列のすべての要素の値は0(文字型の場合はヌルストリング)に設定されます．
- すでに，DIM文で配列宣言されている配列変数を再度DIM文で配列宣言すると，"Duplicate Definition"エラーになります．このような場合は，ERASE文で配列宣言を取り消した後，DIM文を実行してください．

例　DIM MT(3,3,3)

- 3×3×3次元の配列変数MTの配列宣言を行います．この場合，添字の最大値が10以下なので配列宣言の必要はないのですが，メモリの節約のためにDIM文を実行します．

プログラム例

```
100 ' DIM ｻﾝﾌﾟﾙ
110 DIM KU(9,9)
120 FOR I=1 TO 9:FOR J=1 TO 9
130   KU(I,J)=I*J
140   PRINT USING "####";KU(I,J);
150 NEXT J:PRINT:NEXT I
160 END
Ok
run
   1   2   3   4   5   6   7   8   9
   2   4   6   8  10  12  14  16  18
   3   6   9  12  15  18  21  24  27
   4   8  12  16  20  24  28  32  36
   5  10  15  20  25  30  35  40  45
   6  12  18  24  30  36  42  48  54
   7  14  21  28  35  42  49  56  63
```

```
     8   16   24   32   40   48   56   64   72
     9   18   27   36   45   54   63   72   81
Ok
```

- 九九の表を出力します.
- 110行で9×9の2次元配列KUを宣言しています.
- 130行で配列KUに九九の値を代入しています. 配列KU(I,J)が持つ値はI×Jになります.
- 140行でPRINT USING文を使って, 4桁右詰めで数値を出力し, 九九の表を作ります.

参照：ERASE, OPTION BASE

DSKF

ディスク・エフ：disk information

機能　フロッピーディスクの各種情報を返す

書式　DSKF(〈ドライブ番号〉〔,〈機能〉〕)

解説

・〈ドライブ番号〉で指定されたドライブ，およびそこに入っているフロッピーディスクに関する情報を関数の値として返します．

・〈機能〉は値として返す情報の種類を指定します．〈機能〉として指定する値と，そのときに返される関数の値の意味は次のようになっています．

省略したとき：指定したドライブに入っているフロッピーディスクの残り容量をクラスタ数で返す．

指定したとき：指定したドライブの下記の情報を返す．

0：最大トラック番号(＝片面当たりのトラック数－1)
1：1トラック当たりのセクタ数
2：片面フロッピーディスク(0を返す)あるいは両面フロッピーディスク(1を返す)
3：1トラック当たりのクラスタ数
4：フロッピーディスク1枚当たりのクラスタ数
5：ディレクトリのあるトラック番号
6：1クラスタ当たりのセクタ数
7：FATの開始セクタ番号
8：FATの終了セクタ番号
9：FATの数
10：IDのあるセクタ番号

・ドライブの種類やディレクトリ，クラスタ，FATについては，**プログラマーズガイド**第15章「フロッピーディスクとファイルの構造」を参照してください．

例

DSKF(1)　返される値⇒(ドライブ1に入っているフロッピーディスクの残り容量)

DSKF(2,0)　返される値⇒(ドライブ2の最大トラック番号)

・5.25インチ両面高密度ミニフロッピーディスクのとき76を返します．
・5.25インチ両面倍密度ミニフロッピーディスクのとき39を返します．

```
100 ' DSKF ｻﾝﾌﾟﾙ
110 INPUT "ドライブ番号";DR
120 PRINT "トラック  =";DSKF(DR,0);"/ドライブ"
130 PRINT "セクタ    =";DSKF(DR,1);"/トラック"
140 PRINT "ディスクの残り容量 =";DSKF(DR)
150 END
Ok
```

• 5.25インチ両面高密度フロッピーディスクの場合

```
run⏎
ドライブ番号? 2⏎
トラック  = 76 /ドライブ
セクタ    = 26 /トラック
ディスクの残り容量 = 148
Ok
```

• 5.25インチ両面倍密度フロッピーディスクの場合

```
run⏎
ドライブ番号? 2⏎
トラック  = 39 /ドライブ
セクタ    = 16 /トラック
ディスクの残り容量 = 43
Ok
```

• 指定されたドライブの最大トラック数，1トラック当たりのセクタ数およびフロッピーディスクの残り容量を出力します．

• 実行例の数値はシステム構成などによって異なることがあります．

DSKI$

ディスク・アイ・ダラー：disk input $

機能 フロッピーディスクから直接データを読み込む

書式 DSKI$ (〈ドライブ番号〉,〈サーフェス番号〉,〈トラック番号〉,〈セクタ番号〉)

解説

• フロッピーディスク上の指定したセクタから直接データを読み込みます．

• データは0番バッファに読み込まれるとともに，256文字の文字列を値として返します．

• 文字列の長さは255文字までしか許されませんから，DSKI$関数の値として256バイトのデータすべてを一度に得ることはできません．そこで，データを得るときは，次の方法を用います．

(1) ランダムファイルの場合と同じようにFIELD文により，0番バッファへ複数の文字変数を割り当てる．

(2) VARPTR関数により，0番バッファの置かれているメモリ番地を調べ，その領域をPEEK関数を使って読み出す．

• 〈トラック番号〉,〈セクタ番号〉として指定できる値の範囲は，指定された〈ドライブ番号〉のディスクドライブの種類によって異なりますが，BASICはこれらの値を自動的に調べ，不当であった場合には "Illegal function call" エラーとします．

詳しくは，**プログラマーズガイド** 第15章「フロッピーディスクとファイルの構造」を参照してください．

例 DSKI$(1,0,18,1)

• ドライブ1に入っているフロッピーディスクのサーフェス0，トラック18，セクタ1の内容を0番バッファに読み込みます．

プログラム例

```
100 ' DSKI$ ｻﾝﾌﾟﾙ
11Ø FIELD #Ø,128 AS A$(Ø),128 AS A$(1)
12Ø INPUT "ﾄﾞﾗｲﾌﾞ,ｻｰﾌｪｽ,ﾄﾗｯｸ,ｾｸﾀ";DR,SU,TR,SE
13Ø DU$=DSKI$(DR,SU,TR,SE)
14Ø FOR I=Ø TO 1
15Ø   FOR J=Ø TO 7
16Ø     FOR K=1 TO 16
17Ø       D$=HEX$(ASC(MID$(A$(I),J*16+K,1)))
18Ø       PRINT RIGHT$("Ø"+D$,2);" ";
19Ø     NEXT K:PRINT
2ØØ   NEXT J
21Ø NEXT I
22Ø END
Ok
run ⏎
ﾄﾞﾗｲﾌﾞ,ｻｰﾌｪｽ,ﾄﾗｯｸ,ｾｸﾀ? 1,1,2,4 ⏎
3A 88 EF 6F C1 D1 C9 3A Ø1 87 B7 C8 E5 D5 C5 EB
21 ØA ØØ E5 D5 DD 21 AB 23 CD ØØ 85 7D C6 3Ø 2A
B3 18 CD 42 85 D1 E1 DD 21 ØC 24 CD ØØ 85 7D C6
3Ø 2A B3 18 24 CD 42 85 C1 D1 E1 C9 ØØ ØØ E5 D5
```

```
C5 F5 AF 32 3C 1B 3D 32 3D 1B 3A 01 87 B7 20 06
2A 86 EF 22 98 85 F1 18 0E 3A A2 1C 3C 20 05 3A
A3 1C 18 03 CD 4F 21 CD 74 1C CA 0A 1C FE 7F CA
2B 1C FE 08 CA 2B 1C CD A5 1C 3A 01 87 FE 3D 38
0A 3A A1 1C 47 CD E9 1E C3 F0 1B 11 83 1C 21 01
87 E5 7E 23 4F 06 00 09 06 0A 1A B7 28 0C 79 FE
3D 30 3F 1A 13 77 23 0C 10 F0 E1 71 11 8D 1C 21
80 86 E5 7E 23 4F 06 00 09 09 06 0A 1A B7 28 10
79 FE 3D 30 1D 1A 13 77 23 1A 13 77 23 0C 10 EC
E1 71 3A 01 87 FE 3D 30 17 3A 3C 1B B7 20 2B C3
59 1B AF 32 A2 1C F1 3A A1 1C 47 CD E9 1E 18 E9
AF 32 A2 1C 3A 3C 1B B7 20 10 CD 4F 21 FE 08 28
Ok
```

• 指定されたフロッピーディスクのセクタの内容を表示します．このプログラム例は5.25インチ両面高密度ミニフロッピーディスクのものです．

• 120行でドライブ番号，サーフェス番号，トラック番号，セクタ番号を読み込みます．

• 130行で指定されたセクタの内容を0番のバッファに取り込み，最初の256文字を変数DU$に代入します．しかし文字型の変数は255文字以上は許されないので，最後の1文字は無視されます．

• 170行でバッファの内容を1文字ずつ順に16進数に変換して変数D$に代入し，180行で出力します．

• 代入された16進数が1桁の場合は，左側に0を補って2桁にするために，180行でRIGHT$関数を使います．

• 実行例の数値はフロッピーディスクに書き込まれている内容によって異なります．

注意：• 5.25インチ両面高密度ミニフロッピーディスクのサーフェス0，トラック0は，1セクタ128バイトでフォーマットされています．サーフェス0，トラック0のセクタ番号は奇数番号でのみ指定可能で，2セクタ(256バイト)が一度に読み込まれます．

• 複数のフロッピーディスクが接続されている場合，それぞれのフロッピーディスクの情報についてはDSKF関数によって調べることができます．

参照：DSKO$, FIELD, DSKF, VARPTR, PEEK

DSKO$

ディスク

ディスク・オー・ダラー：disk output $

機能 フロッピーディスクに直接データを書き込む

書式 DSKO$ 〈ドライブ番号〉,〈サーフェス番号〉,〈トラック番号〉,〈セクタ番号〉

解説

• フロッピーディスク上の指定したセクタへ直接データを書き込みます.

• DSKO$文は，そのパラメータにより指定したセクタに0番バッファの内容(256バイト)を書き込みます.

• 書き込むデータは，次の方法で準備します.

(1) ランダムファイルの場合と同じようにFIELD文により，0番バッファに変数領域を割り当てた後，LSET文，RSET文により行う.

(2) VARPTR関数により0番バッファの置かれているメモリ番地を調べ，POKE文によりデータを書き込む.

• 指定する〈ドライブ番号〉のフロッピーディスクが5.25インチ片面倍密度ミニフロッピーディスクのときは，〔,〈サーフェス番号〉〕を省略します(コンマを含め省略).

• 〈トラック番号〉,〈セクタ番号〉として指定できる値の範囲は，指定された〈ドライブ番号〉のディスクドライブの種類によって異なりますが，BASICはこれらの値を自動的に調べ，不当であった場合には "Illegal function call" エラーとします.

例 DSKO$ 1,0,18,2

• ドライブ1に入っているフロッピーディスクのサーフェス0，トラック18，セクタ2に0番バッファの内容(256バイト)を書き込みます.

プログラム例

ディスク

• 5.25インチ両面高密度フロッピーディスクの場合

```
100 ' DSKO$ ｻﾝﾌﾟﾙ
110 FIELD #0,1 AS A$,1 AS B$,254 AS C$
120 LSET A$=CHR$(0)
130 INPUT "ドライブ";DR:IF DR<0 OR 2<DR THEN 130
140 INPUT "ファイル数";H$
150 H=VAL(H$)
160 IF H<0 OR 15<H THEN 140
170 LSET B$=CHR$(H)
180 LINE INPUT "メッセージ? ";M$
190 LSET C$=M$
200 DSKO$ DR,0,35,23
210 END
Ok
run⏎
ドライブ? 2⏎
ファイル数? 4⏎
メッセージ? width 80,25⏎
Ok
```

・フロッピーディスクのIDセクタを書き換えます．このプログラム例は，5.25インチ両面高密度ミニフロッピーディスクの場合です．

・IDセクタは，5.25インチ両面高密度ミニフロッピーディスクの場合サーフェス0，トラック35，セクタ23ですから，200行のようにしてIDセクタに書き込みます．

・5.25インチ両面倍密度フロッピーディスクの場合

```
100 ' DSKO$ ｻﾝﾌﾟﾙ
110 FIELD #0,1 AS A$,1 AS B$,254 AS C$
120 LSET A$=CHR$(0)
130 INPUT "ドライブ";DR:IF DR<0 OR 2<DR THEN 130
140 INPUT "ファイル数";H$
150 H=VAL(H$)
160 IF H<0 OR 15<H THEN 140
170 LSET B$=CHR$(H)
180 LINE INPUT "メッセージ? ";M$
190 LSET C$=M$
200 IF DSKF(DR,2)=0 THEN DSKO$ DR,18,13
                       ELSE DSKO$ DR,1,18,13
210 END
Ok
run⏎
ドライブ? 2⏎
ファイル数? 4⏎
メッセージ? width 80,25⏎
Ok
```

・フロッピーディスクのIDセクタを書き換えます．このプログラム例は，5.25インチ片面倍密度ミニフロッピーディスクと，5.25インチ両面倍密度ミニフロッピーディスクの場合です．

・IDセクタは，5.25インチ片面倍密度ミニフロッピーディスクの場合トラック18，セクタ13で，5.25インチ両面倍密度ミニフロッピーディスクの場合サーフェス1，トラック18，セクタ13ですから，200行のようにしてIDセクタに書き込みます．

注意：・5.25インチ両面高密度ミニフロッピーディスクのサーフェス0，トラック0は，1セクタ128バイトでフォーマットされています．サーフェス0，トラック0のセクタ番号は奇数番号でのみ指定可能で，0番バッファの内容(256バイト)は，指定セクタとその次のセクタにわたって書き込まれます．

・この命令は現在のディスクファイルを壊すおそれがありますので，フロッピーディスクおよびファイルの構成を正しく理解したうえで用いてください．

なお，これについては**プログラマーズガイド** 第15章「フロッピーディスクとファイルの構造」で説明されています．

・複数のフロッピーディスクが接続されている場合，それぞれのフロッピーディスクの情報についてはDSKF関数により調べることができます．

参照：DSKI$, FIELD, DSKF, VARPTR

EDIT

エディット：edit

機能 指定された行を画面上へ表示し，カーソルを移動する

書式 EDIT 〈行番号〉

解説

• 指定された行を表示し，カーソルをその先頭に置きます．

• エラー発生時や，STOP文を実行した直後に "EDIT□." を実行すると，エラーの発生した行またはSTOP文を含む行が表示され，直ちに修正作業に入ることができます．

• EDITコマンド実行中は，[⇧]（ロールアップキー）や[⇩]（ロールダウンキー）により指定された行の前後の表示，編集することもできます．

• Pオプション付きでセーブされたファイルをロードして，EDITコマンドを実行した場合には "Illegal function call" エラーとなります．

• EDITをぬけるには，[画面消去]を押します．

例 EDIT 30

• メモリ中のプログラムの30行目を表示し，カーソルをその先頭に置きます．

実行例

```
edit 130↵
130 ON COM GOSUB *COMMUNICATE
```

• COM ON/OFF/STOPのプログラム例で実行した場合，上記のように130行が表示され，カーソルが先頭にきます．

注意： • EDITコマンド実行中，インサートモードにてプログラムの文字数を追加して[↵]を押した場合，その編集行の最後が画面の右端(WIDTH80だと，80文字目)になっていると，その最後の文字がプログラムから抜ける場合があります．
このような場合，[↵]を押す前に，[コントロール]＋[X]キーでカーソルをプログラムの最後に持ってくるようにすれば，この現象は起こりません．
• CONSOLE文により，テキストのスクロール領域(スクロールウィンドウ)を3行以下にした場合，スクロールウィンドウ以上の文字数を持つ行は編集することができません．
EDITを行うときは，スクロールウィンドウを大きめにとってください．

END

エンド：end

機能 プログラムを終了する

書式 END

解説
• プログラムの実行を終了し，すべてのファイルをクローズしてコマンドレベルに戻ります．
• END文は，プログラムの実行を終了させたいところに置けばよく，また，いくつ置いてもかまいません．
• プログラムの最後のEND文は省略することができます．ただし，この場合，ファイルのクローズは行われません．

例 END
• プログラムを終了する．

プログラム例

```
100 ' END ｻﾝﾌﾟﾙ
110 PRINT "'e'キーを押すと終了します！"
120 A$=INPUT$(1)
130 IF A$="e" THEN END ELSE PRINT A$:GOTO 120
Ok
run ⏎
'e'キーを押すと終了します！
a
b
c
d
Ok
```

• "e"が入力されるまで，入力した文字を出力します．
• 130行で，プログラムを終了させるか，入力された文字を出力して次の入力を待つかを判断しています．
• 実行例では"a"，"b"，"c"，"d"そして"e"を入力しています．

EOF

ディスク

エンド・オブ・ファイル：end of file

機　能　ファイルが終了したかどうかを値で返す

書　式　EOF(〈ファイル番号〉)

解　説

• シーケンシャルファイル(ディスクファイル)において，〈ファイル番号〉で指定されたファイルが終わりに達したかどうかを調べる関数です．

• ファイルが終わりに達したとき，EOF関数は真(−1)，そうでなければ偽(0)を返します．

• ファイル番号で指定されたファイルは，入力モード(INPUTモード)でオープンされていなければなりません．

• 指定されたファイルがRS-232Cコミュニケーションチャネル(COM:)であった場合には，EOF関数は，バッファが空であった場合に値を真とします．

例　EOF(1)　返される値　−1(ファイルが終わりに達している)

0(まだ終わりではない)

• ファイル番号1のファイルが終わりに達していれば"−1"，そうでなければ"0"の値を返します．

プログラム例

ディスク

```
100 ' EOF ｻﾝﾌﾟﾙ
110 OPEN "2:tdata" FOR OUTPUT AS #1
120  FOR I=1 TO 10
130    PRINT #1,I
140  NEXT I
150 CLOSE
160 OPEN "2:tdata" FOR INPUT AS #1
170  IF EOF(1) THEN 200
180  INPUT #1,A:PRINT A
190  GOTO 170
200 CLOSE
210 END
Ok
run
 1
 2
 3
 4
 5
 6
 7
 8
 9
 10
Ok
```

• "tdata"というシーケンシャルファイルを作った後，その内容を読み出して画面に出力します.

• 110行から150行で，1から10までの数をファイルに書き込みます.

• 160行から200行では書き込んだ内容を読み出し，画面に出力を行っています.

• 170行で，ファイルを最後まで読み込んだかどうかを判断しています.

注意：• カセットテープを使用している場合は，EOF関数を使うことはできません.

ERASE

イレーズ：erase

機能 配列変数の宣言を取り消す

書式 ERASE 〈配列変数名〉〔,〈配列変数名〉...〕

解説 • 指定した配列変数の宣言を取り消します．以後，同一変数名の配列変数をDIM文で宣言することができます．

• ERASE文は，配列変数が専有していたメモリ領域を解放します．不要になった配列変数を消すことによって，より効率よくメモリを使うことができます．

• ERASE文を実行せずに同一変数名で宣言すると，"Duplicate Definition"エラーが起こります．

例 ERASE A,B$

• 配列変数AとB$の配列宣言を取り消します．

プログラム例

```
100 ' ERASE ｻﾝﾌﾟﾙ
110 DIM A(100),B(100)
120 PRINT FRE(0):ERASE A:PRINT FRE(0)
130 DIM A(200) :PRINT FRE(0)
140 DIM B(200)          :REM 宣言できません！
150 END
Ok
run⏎
 9512
 9925
 9112
Duplicate Definition in 140
Ok
```

• 一度宣言した配列をERASE文で取り消し，別の大きさで再度宣言を行います．

• 110行で配列A, Bを宣言しています．

• 120行で，配列Aを消去する前と消去した後のメモリの未使用領域の大きさを出力します．

• 130行で配列Aを宣言しなおして，メモリの未使用領域の大きさを出力します．

• 140行で配列Bを宣言しなおそうとしていますが，Bは消去されていないためエラーが発生します．

• 実行例の数値は，システム構成などによって異なります．

参照：DIM

ERL/ERR

エラー・ライン/エラー・ナンバ：error line/error number

機能 エラーが起こったときのエラーの発生した行番号とエラーコードを返す

書式 ERL

ERR

解説 • エラーが発生した時点で，ERL変数はエラーが発生した行番号を，ERR変数はそのエラーコードを持っています．この値を参照することによって，エラーの起こった行番号と，エラーの種類を知ることができます．

• エラーがダイレクトモードで生じた場合は，ERL変数は値として65535を返します．

• 通常，ERL変数およびERR変数は，ON ERROR GOTO文によって指定した処理ルーチンにおいて，処理の流れを制御するために使われます．

例 ERL [返される値] （エラーが発生した行の行番号）

ERR [返される値] （エラーの種類を示すエラーコード）

プログラム例

```
100 ' ERL/ERR ｻﾝﾌﾟﾙ
110 ON ERROR GOTO 160
120 INPUT "x,y";X,Y
130 ANS=X^Y
140 PRINT X;"の";Y;"乗は";ANS;"です。"
150 END
160 'エラー処理
170 IF ERR=11 AND ERL=130 THEN PRINT " ??? 定義されません。":END
180 ON ERROR GOTO 0
Ok
run⏎
x,y? 2,3⏎
 2 の 3 乗は 8 です。
Ok
run⏎
x,y? -1,3⏎
-1 の 3 乗は-1 です。
Ok
run⏎
x,y? 0,-1⏎
 ??? 定義されません。
Ok
```

• 2つの実数x，yを読み込んでx^yを出力します．

• 110行で，エラーが生じたとき160行へ制御を移すように設定します．

• 160行以降のエラー処理ルーチンにおいて，130行でエラーコード11，"Division by zero"エラーが生じたとき，"???　定義されません。"を出力して終わります．その他のエラーの

ときは180行を実行して通常のエラー処理に入ります.

参照：ON ERROR GOTO

ERROR

エラー：error

機能　エラーの発生をシミュレートする

書式　ERROR 〈エラー番号〉

解説

- 〈エラー番号〉の値は1～255までの数値で指定します．
- 指定した値がBASICでエラーコードとして使われている場合は，ERROR文が実行されると，〈エラー番号〉に対応するエラーメッセージを表示し，コマンドレベルに戻ります．また，指定した〈エラー番号〉に対応するエラーメッセージがないときは，"Unprintable error"を表示します．
- ON ERROR GOTO文がある場合，ERROR文が実行されると，エラー処理ルーチンへ分岐します．このとき，ERR変数は指定した〈エラー番号〉の値を持ちます．

例　ERROR 5

- "Illegal function call"エラーを起こします．

プログラム例

```
100 ' ERROR ｻﾝﾌﾟﾙ
110 ON ERROR GOTO 180
120 INPUT "３桁の整数(+)を入力してください:";N
130 IF FIX(N)<>N THEN ERROR 251
140 IF N<100 OR N>999 THEN ERROR 250
150 PRINT USING "<###>";N
160 ON ERROR GOTO 0
170 END
180 IF ERR=250 THEN PRINT " ３桁とは１００から９９９!!!"
190 IF ERR=251 THEN PRINT " 実数は入力できません "
200 PRINT
210 RESUME 120
Ok
run
３桁の整数(+)を入力してください:? 425369
 ３桁とは１００から９９９!!!

３桁の整数(+)を入力してください:? 12.5
 実数は入力できません

３桁の整数(+)を入力してください:? 159
<159>
Ok
```

- 入力された数値のチェックを行い，誤りに対してエラーメッセージを表示します．
- もし実数が入力されたら，130行で251番のエラーを発生させます．
- 100から999の間にない数を入力したら，140行で250番のエラーを発生させます．
- 160行ではプログラムを終了する前に，通常のエラー処理に戻しています．

• 180行以降のエラー処理ルーチンでは，エラーメッセージを出力したあと，RESUME 文を使って120行からプログラムを再スタートさせます．

参照：ON ERROR GOTO, ERL/ERR，資料6 「エラーメッセージ」

EXP

イクスポネンシャル：exponential

機能 e(2.7128)を底とする指数関数の値を返す

書式 EXP(〈数式〉)

解説 • 〈数式〉の値である指数関数の値を返します. ただし, 〈数式〉の値が87.33655より大きい場合には"Overflow"エラーとなります.

• EXP関数の演算は単精度で行われます.

例 EXP(3) 返される値 20.0855

EXP(−12) 返される値 6.14422E−06

EXP(100) 返される値 "Overflow" エラー

プログラム例

```
100 ' EXP ｻﾝﾌﾟﾙ
110 INPUT D
120 HS=(EXP(D)-EXP(-D))/2:HC=(EXP(D)+EXP(-D))/2
130 PRINT "SINH";USING "###";D;:PRINT " = ";HS,
140 PRINT "COSH";USING "###";D;:PRINT " = ";HC;
150 END
Ok
run↵
? 2↵
SINH  2 =  3.62686          COSH  2 =  3.7622
Ok
```

• EXP関数を使ってハイパボリック・サイン(SINH)とハイパボリック・コサイン(COSH)の値を求めます.

• $\sinh\theta=(e^{\theta}-e^{-\theta})/2$

$\cosh\theta=(e^{\theta}+e^{-\theta})/2$

で求めています.

FIELD

フィールド：field

機能 ランダムファイルのバッファに変数領域を割り当てる

書式 FIELD〔#〕〈ファイル番号〉，〈フィールド幅〉AS〈文字変数〉〔，〈フィールド幅〉AS〈文字変数〉...〕

解説
- ファイルバッファをランダムファイルバッファとして使用するために，ファイルバッファの中に文字変数の領域を割り当てます．
- FIELD文は，ランダムファイルの入出力を行う前に実行します．
- 〈ファイル番号〉はOPEN文によってファイルをオープンしたときに指定した番号です．また，特に〈ファイル番号〉が0のときDSKI$関数，DSKO$文において使用されるランダムバッファを定義します．
- 〈フィールド幅〉は〈文字変数〉に割り当てる文字数(バイト数)です．バッファの大きさは256バイトなので，各文字変数の〈フィールド幅〉の合計は256バイト以内でなければなりません．
- 1つのバッファに対して，何回かに分けてFIELD文を実行することができます．2回目以降FIELD文を使用する場合，以前のFIELD文で指定したフィールド幅の合計を仮りのデータとして他の変数に割り当て，そのあとに追加するフィールドを指定してください．
- FIELD文で指定した〈文字変数〉はLSET/RSET文で定義します．また，この変数をINPUT文，LET文の左辺で使用することはできません．もし使用すると，その変数は一般の文字変数領域に登録され，FIELD文でのバッファの割り当てが無効となり，正しいファイルの入出力が行われなくなります．
- FIELD文で指定した変数は，FIELD文を実行した時点で，〈フィールド幅〉で指定した文字数のヌルキャラクタ(キャラクタコードが0のキャラクタ)がセットされます．

例 FIELD #1，128 AS A$，128 AS B$
- ファイル番号1のファイルバッファに，文字変数A$，B$の領域を各128バイトずつ割り当てます．

プログラム例

ディスク

```
100 ' FIELD ｻﾝﾌﾟﾙ
110 OPEN "2:data1" AS #1
120  FIELD#1,15 AS A$,15 AS B$,15 AS C$
130  LSET A$="NEC"
140  LSET B$="Personal"
150  LSET C$="Computer"
160  PUT #1,1
170  GET #1,1
```

```
180  PRINT A$;B$;C$
190 CLOSE
200 END
Ok
run⏎
NEC            Personal        Computer
Ok
```

• "data1"というランダムファイルを作ります.

• 120行で初めの15バイトをA$, 次の15バイトをB$, 次の15バイトをC$にそれぞれ割り当てます.

参照：OPEN, GET, PUT, LSET/RSET

FILES/LFILES

ディスク

ファイルズ/エル・ファイルズ：files/list files

機能 フロッピーディスクに入っているファイルの情報を出力する

書式 (1) FILES〔〈ドライブ番号〉〕

(2) LFILES〔〈ドライブ番号〉〕

解説 •〈ドライブ番号〉で指定されたフロッピーディスク上に登録されているファイルの名前，種類，およびその大きさを出力します.

•書式(1)の場合ディスプレイ画面に，(2)の場合プリンタに出力します.

•〈ドライブ番号〉が省略されたときは，ドライブ1が選択されます.

•ファイルの種類は，出力される〈ファイル名〉と〈拡張子〉との区切り記号が空白の場合には，そのファイルはアスキーセーブされたプログラムファイル，またはデータファイルであることを示します．区切り文字が，ピリオドの場合には，バイナリセーブによって作られたプログラムファイル，アスタリスクの場合にはBSAVEコマンドによって作られた機械語プログラムファイルであることを表します.

•ファイルの大きさは，クラスタを単位として表示されます．5.25インチ両面高密度ミニフロッピーディスクと8インチフロッピーディスクの1クラスタは26セクタ(6656バイト)です．5.25インチ両面倍密度フロッピーディスクの1クラスタは8セクタ(2048バイト)です.

例 FILES 2

•ドライブ2に入っているファイルの情報を画面に表示します.

実行例

ディスク

```
files↵
usrdic.mh  6    udm   *nmh 1    prload.mh  1    font  .mh  6    dfont *nmh 1
cfont *nmh 1    cdcnv .mh  7    cod   *nmh 1    setpdt.j88 1    tdata      1
test1 .    1    sample.key 1    defkey.    1
Ok
```

•ドライブ1に入っているフロッピーディスクの内容を表示します.

参照：SAVE, BSAVE

FIX

フィックス：fix

機能　数値の整数部の値を返す

書式　FIX(〈数式〉)

解説

- 〈数式〉の絶対値の小数点以下を切り捨て，〈数式〉の符号をつけた値を返します．
- FIX関数とINT関数の値は〈数式〉の値が正のとき等しくなりますが，負のときは値が異なります．(例：INT(−3.5)＝−4, FIX(−3.5)＝−3)

例

FIX(1.23) 返される値 ⇒ 1

FIX(−3.2) 返される値 ⇒ −3

プログラム例

```
100 ' FIX ｻﾝﾌﾟﾙ
110 FOR A=-2 TO 2 STEP .5
120   PRINT "A =";A,"FIX(A) =";FIX(A),"INT(A) =";INT(A)
130 NEXT A
140 END
Ok
run⏎
A =-2          FIX(A) =-2    INT(A) =-2
A =-1.5        FIX(A) =-1    INT(A) =-2
A =-1          FIX(A) =-1    INT(A) =-1
A =-.5         FIX(A) = 0    INT(A) =-1
A = 0          FIX(A) = 0    INT(A) = 0
A = .5         FIX(A) = 0    INT(A) = 0
A = 1          FIX(A) = 1    INT(A) = 1
A = 1.5        FIX(A) = 1    INT(A) = 1
A = 2          FIX(A) = 2    INT(A) = 2
Ok
```

- いくつかの実数に対して，FIX関数とINT関数を実行した結果を出力します．
- FIX関数は小数点以下を切り捨てます．
- INT関数は，たとえばA＝−1.5のとき，−1.5を超えない最大の整数になりますからINT(−1.5)＝−2になります．

参照：INT

FOR...TO...STEP～NEXT

フォー・トゥー・ステップ・ネクスト：for...to...step～next

機能　一連の命令を指定回数だけ繰り返し実行する

書式　FOR〈変数〉=〈初期値〉TO〈終値〉〔STEP〈増分〉〕

　～

NEXT〔〈変数〉〔,〈変数〉...〕〕

解説

・FOR文とNEXT文ではさまれた一連の命令を指定した回数だけ繰り返し実行します．回数の指定は，〈初期値〉，〈終値〉，〈増分〉で決まります．

・〈変数〉には整数変数，単精度変数を使用します．文字変数，倍精度変数は使用できません．

・〈初期値〉，〈終値〉，〈増分〉は数式です．

・FOR～NEXTの動作は次のようになります．

(1)　〈変数〉に〈初期値〉が設定されます．

(2)　FOR以後からそれに対応するNEXTまで実行されます．

(3)　STEPによって指定された〈増分〉を〈変数〉に加え，それを新しい〈変数〉とします．

(4)　〈変数〉の値が〈終値〉と比べられ，〈終値〉に達していなければ(2)へ移り，同じ処理が繰り返されます．

これをFOR～NEXTループと呼びます．

・STEPが省略された場合〈増分〉は1とみなされます．また〈増分〉は負の値をとることもできます．このとき〈初期値〉は，〈終値〉より大きくしなければなりません．この場合〈変数〉が〈終値〉よりも小さくなるまでFOR～NEXTループが繰り返されます．

・次の場合FOR～NEXTループは実行されず，NEXT文の次の文に実行が移ります．

(1)　〈増分〉が正の値で，〈初期値〉が〈終値〉より大きい場合．

(2)　〈増分〉が負の値で，〈初期値〉が〈終値〉より小さい場合．

(3)　〈初期値〉と〈終値〉が同じ値で〈増分〉が0の場合．

ただし，〈変数〉には〈初期値〉が代入されています．

・FOR～NEXTループは入れ子構造にすることができます．これは，1つのFOR～NEXTループ中にもう1つのFOR～NEXTループを置くことができるということです．このとき，それぞれの〈変数〉は別のものを使います．また，1つのFOR～NEXTループは完全に他のFOR～NEXTループの内部になければなりません．

・FOR文とNEXT文とは必ず1対1に対応していなければなりません．したがって，

IF 文中に NEXT 文を置くことができない場合があります.

例　FOR I＝1 TO 10：PRINT "＊"；：NEXT I

• PRINT "＊"；を10回実行します．すなわち，"＊"を10個表示します．

プログラム例

```
100 ' FOR...TO...STEP ~ NEXT ｻﾝﾌﾟﾙ
110 FOR I=1 TO 4
120   PRINT "STEP";I
130   FOR J=1 TO 15 STEP I
140     PRINT J,
150   NEXT J
160   PRINT
170 NEXT I
180 END
Ok
run
STEP 1
 1             2             3             4             5
 6             7             8             9             10
 11            12            13            14            15

STEP 2
 1             3             5             7             9
 11            13            15
STEP 3
 1             4             7             10            13

STEP 4
 1             5             9             13
Ok
```

• 1から15までの数値を増分を＋1，＋2，＋3，＋4として出力します．

• 120行から160行までが4回繰り返し実行され，さらにその中で140行を繰り返し実行しています．

参照：WHILE～WEND

FPOS

エフ・ポジション：file position

機能 ファイル中での物理的な位置を返す

書式 FPOS(〈ファイル番号〉)

解説 ・〈ファイル番号〉によって指定されたファイルが最後に読み書きをした位置を返します．

・指定されたファイルがディスクファイルの場合には，最後に読み書きしたフロッピーディスク上の物理的な位置を返します．この値は，トラック0，サーフェス0，セクタ1を0としたときの通し番号です．

・5.25インチ両面高密度ミニフロッピーディスクの場合，通し番号は次のように割り当てていきます．

サーフェス	0	0	0	···	0	0	1	1	1	···	1	1	0	0	0	···
トラック	0	0	0	···	0	0	0	0	0	···	0	0	1	1	1	···
セクタ	1	2	3	···	25	26	1	2	3	···	25	26	1	2	3	···
通し番号	0	1	2	···	24	25	26	27	28	···	50	51	52	53	54	···

・5.25インチ両面倍密度ミニフロッピーディスクの場合，通し番号は次のように割り当てていきます．

サーフェス	0	0	0	···	0	0	1	1	1	···	1	1	0	0	0	···
トラック	0	0	0	···	0	0	0	0	0	···	0	0	1	1	1	···
セクタ	1	2	3	···	15	16	1	2	3	···	15	16	1	2	3	···
通し番号	0	1	2	···	14	15	16	17	18	···	30	31	32	33	34	···

・指定されたファイルがプリンタであった場合には，プリンタのヘッド位置を返します．この値はLPOS関数が返す値と同じです．

例 FPOS(2)

・ファイル番号2のファイル中での現在位置を返します．指定されたファイルがディスクファイルの場合，たとえば物理的な位置がトラック2，サーフェス0，セクタ4(5.25インチ両面高密度ミニフロッピーディスクの場合)のときは，107(＝52×2(トラック)＋26×0(サーフェス)＋4(セクタ)－1)を返します．
(5.25インチ両面倍密度ミニフロッピーディスクの場合)のときは，67(＝32×2(トラック)＋16×0(サーフェス)＋4(セクタ)－1を返します．

プログラム例

ディスク

```
100 ' FPOS ｻﾝﾌﾟﾙ
110 OPEN "2:fdata" FOR OUTPUT AS #1
120  PRINT FPOS(1):PRINT
130  FOR I=1 TO 5
140    PRINT #1,STRING$(126,"1")
150    PRINT #1,STRING$(126,"1")
160    PRINT FPOS(1)
170  NEXT I
180 CLOSE
190 END
Ok
run⏎
 144

 146
 147
 148
 149
 150
Ok
```

- ファイルにデータを書き込みながら，フロッピーディスク上の書き込んだ位置を出力します．
- 120行で，ファイルのオープンで読み書きしたフロッピーディスクの位置を出力します．
- 140行，150行で256バイトのデータ，つまり1セクタ分のデータを書き込んでいます．
- 160行で，フロッピーディスク上のデータを書き込んだ位置を出力します．

参照：LOC, LPOS

FRE

フリー：free

機能 メモリの未使用領域の大きさを返す

書式 FRE(〈機能〉)

解説 •〈機能〉は0，1，2の値をとり，それぞれの場合に返される関数値の意味は次のとおりです．

0：未使用変数領域のバイト数．単純変数，配列およびストリングの領域として使用可能な領域の残りバイト数です．

1：未使用テキスト領域のバイト数．プログラムテキストを入れるための領域の残りバイト数です．

2：未使用変数領域と未使用テキスト領域の和のバイト数．

•〈機能〉が0または2のとき，FRE関数は文字列領域の整理のための"ちり集め"処理を必ず行いますので，関数が値を返すまでに時間がかかることがあります．

例 FRE(0) 返される値 (未使用変数領域の大きさ)

FRE(1) 返される値 (未使用テキスト領域の大きさ)

FRE(2) 返される値 (未使用変数領域および未使用テキスト領域の大きさ)

プログラム例

```
100 ' FRE ｻﾝﾌﾟﾙ
110 CLEAR
120 PRINT "変数の未使用領域は";FRE(0);"バイトです"
130 PRINT "テキストの未使用領域は";FRE(1);"バイトです"
140 PRINT "変数およびテキストの未使用領域は";FRE(2);"バイトです"
150 END
Ok
run
変数の未使用領域は 11872 バイトです
テキストの未使用領域は 32578 バイトです
変数およびテキストの未使用領域は 44450 バイトです
Ok
```

• CLEAR文を実行してから変数とテキストの未使用領域の大きさを出力します．

• 実行例の数値は，システム構成によって異なります．

参照：CLEAR

GET

ゲット：get

機　能　ファイル中のデータをバッファに読み込む

書　式　GET〔#〕〈ファイル番号〉〔,〈数〉〕

解　説

・〈ファイル番号〉で指定されたファイル中のデータを，対応するバッファの中に読み込みます．

・指定されたファイルがディスクファイルの場合は，ランダムファイルからの読み込みとして働きます．この場合，指定されたファイルはランダムアクセスのモードでオープンされていなければなりません．

・〈数〉はファイルのレコード番号として解釈され，指定されたレコード(256バイト)がバッファに読み込まれます．レコード番号が省略された場合には，直前に行われたGET文，PUT文で指定されたレコードの次のレコードが読み込まれます．

・指定されたファイルがディスクファイルでない場合，GET文は指定されたファイルに対応する装置から，指定した文字数をバッファの中に読み込みます．この場合，指定されたファイルは入力モードでオープンされていなければなりません．

・〈数〉はファイルからバッファに読み込む文字数を意味し，0から255までの値を指定します．〈数〉が省略されたとき，および0が指定されたときは256文字を読み込みます．

・GET文で読み込んだデータを参照するには，あらかじめFIELD文で割り当てられた変数を使います．

例　GET #3,5

・ファイル番号3でオープンしたランダムファイルの5番目のレコードをバッファに読み込みます(ディスクファイルの場合)．

プログラム例

ディスク

```
100 ' GET ｻﾝﾌﾟﾙ
110 OPEN "2:address" AS #1
120  FIELD #1,30 AS A$,20 AS B$,50 AS C$
130  FOR I=1 TO LOF(1)
140    GET #1,I
150    PRINT "名前",A$
160    PRINT "電話",B$
170    PRINT "住所",C$
180  NEXT I
190 CLOSE
200 END
Ok
run
名前            相川　房子
電話            03-123-4567
住所            東京都　日本橋
```

```
名前                土屋　秀美
電話                Ø43-123-4567
住所                神奈川県　川崎市
Ok
```

• PUT文のプログラム例で作ったデータファイルを読み込んで画面に出力します.

• LOF関数でデータの個数を調べ，130行から180行の間ですべてのデータをGET文で読み込み出力します.

注意：• GET文を実行した後，INPUT#文，LINE INPUT#文等で同一の〈ファイル番号〉を指定すると，そのバッファに入っているデータから入力が行われますので注意してください.

参照：OPEN, FIELD, PUT

G

GET@

ゲット・アットマーク：get @

機 能 グラフィック画面のグラフィックパターンを配列に読み込む

書 式 GET〔@〕(Sx₁,Sy₁)－ | (Sx₂,Sy₂) / STEP(x,y) | ,〈配列変数名〉〔(〈要素ナンバ〉)〕

解 説 •スクリーン座標の2点(Sx₁,Sy₁)と(Sx₂,Sy₂)を対角とする長方形の領域を，〈配列変数名〉で指定された配列変数に読み込みます．

•〈配列変数名〉は，グラフィックパターンを読み込む数値型の配列変数の名前です．また，この配列変数に対して〈要素ナンバ〉を指定することもできます．この〈要素ナンバ〉は，グラフィックパターンを配列変数に読み込む際に，どの要素から格納し始めるかを指定するものです．たとえばDIM A%(10)と指定し，要素ナンバを3(GET(X₁,Y₁)－(X₂,Y₂),A%(3))とすると，A%(1)とA%(2)には以前の値が残り，A%(3)からA%(10)にグラフィックパターンを読み込みます．省略した場合は，配列の最初(〈要素ナンバ〉は0)から格納します．

•GET@文を実行する前に，この配列変数に必要な大きさをDIM文で確保しておきます．この配列変数の大きさは，読み込む領域の大きさ，画面モード，配列変数の型によって異なります．

•1つの配列に1つのパターンしか読み込まない場合は，次の計算式で求めることができます．複数のパターンの場合は，それぞれのパターンについての大きさを合計し，その分確保しなければなりません．

〈必要なバイト数〉＝((〈横のドット数〉＋7)¥8)＊〈縦のドット数〉＊M＋4

白黒モードのとき………M＝1

カラーモードのとき……M＝3

〈添字の値〉＝〈必要なバイト数〉¥N＋1

整数型配列のとき………N＝2

単精度型配列のとき……N＝4

倍精度型配列のとき……N＝8

この〈添字の値〉とは，一次元配列で添字の底が0のとき(OPTION BASE文参照)の最小値です．

•GET@文はPUT@文と密接な関係にあり，通常ペアで使用されます．

•GET@文の実行後，LPは(Sx₂, Sy₂)に移動します．

例 GET@(0,0)－(15,15),G%

・スクリーン座標(0, 0)～(15, 15)の範囲にあるグラフィックパターンを配列変数G%に読み込みます.

プログラム例

```
100 ' GET@ ｻﾝﾌﾟﾙ
110 SCREEN 0,0:CLS 3
120 XD=40:YD=20
130 BYTE=((XD+7)¥8)*YD*3+4
140 FACT=BYTE¥2+1
150 DIM G%(FACT)
160 CIRCLE(XD/2-1,YD/2-1),YD/2,6
170 PAINT(XD/2-1,YD/2-1),6,6
180 GET@(0,0)-STEP(XD-1,YD-1),G%
190 FOR X=0 TO 500 STEP 100
200   PUT@(X,100),G%:PUT@(X,100),G%
210 NEXT X
220 END
Ok
```

・画面左上に黄色の円を描き，それと同じ円が左から右に移動します.

・130行，140行で配列宣言で必要な添字の数を計算しています.

・180行で160行，170行で描いた円を配列に取り込みます.

・190行から210行で取り込んだ円を，6ヶ所に表示します.

・2回PUT@文を実行していますが，この場合機能がXORなので，2回目を実行すると画面がもとの状態に戻ります．つまり，一度描いた円を消去することになります.

注意：・配列変数は数値型の一次元配列でなければなりません.

・GET@文が必要とする配列サイズを，必ず確保しなければなりません.

参照：PUT@, OPTION BASE

GOSUB～RETURN

ゴー・サブ・リターン：go to subroutine～return

機能 サブルーチンの呼び出し，およびサブルーチンからリターンを行う

書式 GOSUB 〈行番号〉

～

RETURN〔〈行番号〉〕

解説 •〈行番号〉から始まるサブルーチンプログラムをGOSUB文で呼び出し，サブルーチンプログラム内のRETURN文によって，GOSUB文の次の文へ実行を移します．

• 1つのサブルーチンの中から他のサブルーチンを呼び出すこともでき，これをサブルーチンの多重化(ネスティング)と呼びます．この多重化はメモリのスタック領域の容量が許す限り行うことができます．もしスタック領域が足りなくなった場合には，"Out of memory" エラーとなります．

• RETURN文はGOSUB文と正しく対応していなければなりません．もし単独で実行した場合，"RETURN without GOSUB" エラーとなります．

• 1つのサブルーチン内に複数のRETURN文を使用する場合は，GOSUB文と正しく対応させなければいけません．

• RETURN文で〈行番号〉を指定することによって，特定の行へ制御を戻すことができます．

例 GOSUB 1000

• 1000行から始まるサブルーチンを呼び出します．

RETURN 100

• サブルーチンから抜け出し，100行へ実行を移します．

プログラム例

```
100 ' GOSUB ~ RETURN ｻﾝﾌﾟﾙ
110 INPUT "底辺 :";BA
120 INPUT "高さ :";H
130 GOSUB 160
140 PRINT "面積は";AR
150 END
160 AR=BA*H/2
170 RETURN
Ok
run⏎
底辺 :? 78⏎
高さ :? 8⏎
面積は 312
Ok
```

• 三角形の底辺と高さを読み込んで，面積を出力します．

• 160行と170行が，面積を計算するサブルーチンです．

注意：• "GO" と "SUB" の間に，最大１個の空白を入れることが許されます("GO" と "SUB"を続けて書いてもかまいません)．

GOTO/GO TO

ゴートゥー：go to

機能 指定した行へ制御を移す

書式 GOTO 〈行番号〉またはGO TO 〈行番号〉

解説
- 〈行番号〉で指定した行へ制御を移します．つまり，〈行番号〉で指定した行から実行を続けます．
- "GO" と "TO" の間に 1 文字のスペースを置くことができます．

例 GOTO 1000

- 1000行へ制御を移します．

プログラム例

```
100 ' GOTO ｻﾝﾌﾟﾙ
110 PRINT "How do you do?"
120 GOTO 110
Ok
run⏎
How do you do?
How do you do?
How do you do?
How do you do?
How do you do?
How do you do?
How do you do?
How do you do?
^C
Break in 110
Ok
```

- [停止] を押すまで "How do you do?" を続けて出力します．
- 実行例では 8 回ループしたところで [停止] が押され，110行でプログラムが終了しています．

HELP ON/OFF/STOP

ヘルプ・オン/オフ/ストップ：**help on/off/stop**

機能 [説明] による割り込みの許可，禁止，停止を設定する

書式

1) HELP ON

2) HELP OFF

3) HELP STOP

解説

- [説明] を押したときに発生する割り込みを，許可(ON)するか，禁止(OFF)するか，停止(STOP)するかを設定します．

  1) 割り込み動作を許可します．この命令実行後は，[説明] を押すごとに割り込みを発生し，ON HELP GOSUB文によって定義されたルーチンに分岐します．

  2) 割り込み動作を禁止します．この命令実行後は，[説明] を押しても処理ルーチンへの分岐は起こりません．

  3) 割り込み動作を禁止します．この命令実行後は，[説明] を押しても押されたことを覚えているだけで，処理ルーチンへの分岐は起こりません．しかし，その後，HELP ON文実行により割り込みが許可されると，先ほどの [説明] を押したことによって処理ルーチンに分岐します．

例 HELP OFF

- [説明] による割り込みを禁止します．

プログラム例

```
100 ' HELP ON/OFF/STOP ｻﾝﾌﾟﾙ
110 ON HELP GOSUB *MESSAGE
120 HELP ON
130 FOR I=1 TO 3
140   INPUT N
150   PRINT "N =";N,"N*N =";N*N
160 NEXT I
170 HELP OFF:END
180 *MESSAGE
190  HELP OFF
200  PRINT:PRINT "数字を入力してください!!"
210  HELP ON
220 RETURN 140
Ok
run⏎
? 2⏎
N = 2          N*N = 4
? [説明]
数字を入力してください!!
? 5⏎
N = 5          N*N = 25
? 9⏎
N = 9          N*N = 81
Ok
```

• 入力された数値の２乗を出力するプログラムです．

• 140行のINPUT文の入力待ちのときに [説明] を押すと，割り込みが発生して180行に分岐します．

• 180行からのサブルーチンでは，"数字を入力してください!!"を出力して，140行に戻ります．

参照：ON HELP GOSUB, KEY ON/OFF/STOP

HEX$

ヘキサ・ダラー：hexadecimal $

機能 10進数を16進数の文字列に変換した値を返す

書式 HEX$(〈数式〉)

解説
- 〈数式〉の値を16進数に変換して，その文字列を返します．
- 〈数式〉の値の範囲は，－32768～65535です．〈数式〉に小数点が含まれる場合は，小数点以下を四捨五入したあと変換します．

例

HEX$(－32768) 返される値 "8000"

HEX$(5000) 返される値 "1388"

HEX$(65535) 返される値 "FFFF"

プログラム例

```
100 ' HEX$ ｻﾝﾌﾟﾙ
110 PRINT "     -0 -1 -2 -3 -4 -5 -6 -7 -8 -9"
120 FOR I=0 TO 9
130   PRINT STR$(I);"- ";
140   FOR J=0 TO 9
150     PRINT RIGHT$("0"+HEX$(I*10+J),2);" ";
160   NEXT J
170   PRINT
180 NEXT I
190 END
Ok
run
    -0 -1 -2 -3 -4 -5 -6 -7 -8 -9
 0- 00 01 02 03 04 05 06 07 08 09
 1- 0A 0B 0C 0D 0E 0F 10 11 12 13
 2- 14 15 16 17 18 19 1A 1B 1C 1D
 3- 1E 1F 20 21 22 23 24 25 26 27
 4- 28 29 2A 2B 2C 2D 2E 2F 30 31
 5- 32 33 34 35 36 37 38 39 3A 3B
 6- 3C 3D 3E 3F 40 41 42 43 44 45
 7- 46 47 48 49 4A 4B 4C 4D 4E 4F
 8- 50 51 52 53 54 55 56 57 58 59
 9- 5A 5B 5C 5D 5E 5F 60 61 62 63
Ok
```

- 0から99までの10進数を，16進数に変換した表を出力します．
- 150行では16進数の値が1桁の場合，上位の桁に0を付けて2桁で表示させています．

参照：OCT$

IF...THEN...ELSE/IF...GOTO...ELSE

イフ・ゼン・エルス/イフ・ゴートゥー・エルス：if...then...else/if...goto...else

機能　論理式の条件判断を行い，次に実行する文を選択する

書式　IF 〈論理式〉 { THEN { 〈文〉 | 〈行番号〉 } | GOTO 〈行番号〉 } 〔ELSE { 〈文〉 | 〈行番号〉 }〕

解説

・論理式の条件によってプログラムの実行を制御します．

・〈論理式〉が真(0以外)ならばTHENまたはGOTO以降を実行し，偽(0)ならばELSE以降が実行されます．もしELSE以降が省略されている場合，次の行が実行されます．

・IF...THEN...ELSE文において，THENやELSEに続けて別のIFを置いて多重にすることができます．

例　IF A$="y" THEN 200 ELSE END

・文字変数A$が"y"のとき200行へ制御が移り，それ以外のときはプログラムの実行を終了します．

プログラム例

```
100 ' IF...THEN...ELSE/IF...GOTO...ELSE ｻﾝﾌﾟ
110 INPUT A
120 FL=0
130 IF A<0 THEN A=ABS(A):FL=1
140 PRINT SQR(A);
150 IF FL THEN PRINT "i" ELSE PRINT
160 END
Ok
run⏎
? 5⏎
 2.23607
Ok
run⏎
? -10⏎
 3.16228 i
Ok
run⏎
? -25⏎
 5 i
Ok
```

・単精度実数を読み込んで，その平方根を出力します．

・負の数の場合130行で，その絶対値をとり，変数FLを1にしておきます．

・150行でFLが1，つまり負の数が入力されたときは答えが虚数なので，"i"を出力しています．

参照：GOTO/GO TO

INKEY$

インキー・ダラー：input key $

機能 キーが押されていればその文字を，キーが押されなければヌルストリングを返す

書式 INKEY$

解説

• キーボードバッファが空であれば，INKEY$関数はヌルストリングを返します．キー入力によりキーボードバッファが空でない場合には，バッファの先頭から1文字取り出し，その文字を返します．

• キー入力された文字は画面に表示されません．

• INKEY$関数で [コントロール] + [C]， [停止] の入力を知ることはできません．INKEY$関数を含むステートメント実行中に [停止] を押すと，プログラムの実行がとまるためです．

例 INKEY$ [返される値] (押されたキーの文字)

プログラム例

```
100 ' INKEY$ ｻﾝﾌﾟﾙ
110 PRINT "何か、キーを押すと終了します。"
120 A$=INKEY$
130 IF A$="" THEN 110 ELES END
Ok
run⏎
何か、キーを押すと終了します。
何か、キーを押すと終了します。
何か、キーを押すと終了します。
何か、キーを押すと終了します。
何か、キーを押すと終了します。
何か、キーを押すと終了します。
何か、キーを押すと終了します。
何か、キーを押すと終了します。
Ok
```

• 何かキーを押すまで，"何か，キーを押すと終了します。"を出力します。

• 実行例では8回"何か，キーを押すと終了します。"を出力したところで，あるキーを押しています．

INP

アイ・エヌ・ピー：input

機能 入力ポートから得た値を返す

書式 INP（〈I/Oアドレス〉）

解説 • 〈I/Oアドレス〉で指定された入力ポートから8ビットのデータを読み込み，それを関数の値として返します．

• 〈I/Oアドレス〉として指定できる値の範囲は0～255です．

例 INP（&H33） 返される値 （33Hポートからの入力データ）

プログラム例

```
100 ' INP ｻﾝﾌﾟﾙ
110 PRINT "'e'キーを押すと終了します。"
120 IF INP(2)=223 THEN END
130 DM$=INPUT$(1):PRINT DM$
140 GOTO 120
Ok
run⏎
'e'キーを押すと終了します。
j
i
O
l
i
j
e
Ok
```

• "e"のキーを押すまで，押したキーの文字を出力し続けます．

• 120行は入力ポートの2番の値が，2進数で11011111になっているかどうかを判断しています．これは"e"のキーが押されたかどうかの判断です．

• 実行例では，"j"，"i"，"o"，"l"，"i"，"j"，"e"の順に入力しています．

参照：OUT

INPUT

インプット：input

機能　キーボードから入力したデータを指定した変数に代入する

書式　INPUT〔"〈プロンプト文〉"｜;｜〕〈変数名〉〔,〈変数名〉...〕
｜,｜

解説　• INPUT文が実行されると，プログラムはキーボードからの入力待ち状態になります．〈プロンプト文〉のあとにセミコロン(;)が続いた場合，さらに疑問符(?)と1文字のスペースを画面に表示します．コンマ(,)が続いた場合，〈プロンプト文〉のあとには何も表示されません．

• データは⏎を押すことによって入力され，変数に代入されます．〈変数名〉をコンマで区切って複数個指定した場合には，入力するデータもコンマで区切って〈変数名〉の数だけ入力します．

• 〈変数名〉と入力されたデータの個数や型が一致しなかったときは，"?Redo from start" と表示し，再び入力待ちになります．

• ⏎だけを押した場合，0あるいはヌルストリングが入力されたとみなされます．この場合も〈変数名〉が複数のときは，必要な数のコンマを入力します．

• 文字変数に文字列を代入する場合，コンマや文字列の前後の意味のある空白を入力したいときは，ダブルクォーテーション(")で文字列を囲みます．

• 値としてダブルクォーテーション(")を入力することはできません．

例　INPUT "DATA＝";D$

• "DATA=? "と表示し，文字列の入力を待ちます．文字列が入力されると文字変数D$に代入します．

プログラム例

```
100 ' INPUT ｻﾝﾌﾟﾙ
110 INPUT "名前 ";NA$
120 INPUT "性別 ";SE$
130 INPUT "年齢 ";OD
140 PRINT "名前  :",NA$
150 PRINT "性別  :",SE$
160 PRINT "年齢  :",OD
170 END
Ok
run⏎
名前 ? 早川  かおる⏎
性別 ? 女⏎
年齢 ? はたち⏎
?Redo from start
年齢 ? 20⏎
名前  :        早川  かおる
```

```
性別　:　　　　　女
年齢　:　　　　　 2Ø
Ok
```

・名前，性別，年齢を読み込んで，それを出力します．

・実行例では年齢を入力するときに数値を入力しなかったために，"?Redo from start" というメッセージが表示され，再び入力待ちになっています．

参照：LINE INPUT

INPUT#

インプット・シャープ：input ＃

機能 シーケンシャルファイルからデータを読み込む

書式 INPUT ＃〈ファイル番号〉,〈変数〉〔,〈変数〉...〕

解説
- データを読み込む対象がシーケンシャルファイルであること，疑問符(?)が表示されないことを除けば，INPUT 文とほぼ同じです．
- 〈ファイル番号〉は，OPEN 文によってオープンしたときに指定したファイル番号です．

例 INPUT ＃1, DT

- ファイル番号が1のシーケンシャルファイルから数値データを読み込み，変数DTに代入します．

プログラム例

ディスク

```
100 ' INPUT# ｻﾝﾌﾟﾙ
110 OPEN "2:Ldata" FOR INPUT AS #1
120  IF EOF(1) THEN 160
130  INPUT #1,A$
140  PRINT A$
150  GOTO 120
160 CLOSE
170 END
Ok
run⏎
BASIC
FORTRAN
COBOL
FORTH
Prolog
LOGO
Ok
```

- PRINT＃文のプログラム例を実行することによって作成されたシーケンシャルファイルの内容を出力します．
- 120行では，ファイルが終わりに達していたら実行を停止するようになっています．

参照：LINE INPUT＃, OPEN, PRINT＃, PRINT＃ USING, WRITE＃

INPUT$

インプット・ダラー：input $

機能 指定されたファイルより指定された長さの文字を返す

書式 INPUT$ (〈文字数〉〔,〔#〕〈ファイル番号〉〕)

解説 •〈ファイル番号〉によって指定されたファイルから，〈文字数〉分の文字列を読み出します．〈ファイル番号〉が省略された場合には，キーボードから入力が行われますが，INPUT 文と異なり，入力された文字はスクリーンには表示されません．

•INPUT$関数は，指定された〈文字数〉の文字が入力されるのを待ち続けますが，すでにバッファに入力済みのデータがある場合には，バッファ中から文字を拾ってきます．INPUT$関数は [停止]，[コントロール] + [C] を除くすべての文字をそのまま読み出しますので，INPUT 文やLINE INPUT 文では入力することのできない改行文字等も入力することができます．

例 INPUT$(5) 返される値 ⇒ (キーボードから5文字読み込んだ文字列)

プログラム例

```
100 ' INPUT$ ｻﾝﾌﾟﾙ
110 A$=INPUT$(1)
120 IF A$=CHR$(&H1B) THEN END
130 IF A$=CHR$(13) THEN PRINT ELSE PRINT A$;
140 GOTO 110
Ok
run↵
Personal
Ok
```

• [ESC] が押されるまで，キーボードから入力した文字を表示します．

• 110行で，キーボードから1文字読み込みます．

• 120行は，入力した文字が [ESC] かどうか判断しています．

• [↵]を入力した場合，現在文字が表示されている行の先頭にカーソルが移動してしまうので，130行で改行するようにします．

• 実行例では "P", "e", "r", "s", "o", "n", "a", "l" と入力したあとに，[ESC] を入力しています．

参照：INPUT, LINE INPUT

INPUT WAIT

インプット・ウェイト：input wait

機 能 キーボードからの入力を時間制限します

書 式 INPUT WAIT 〈待ち時間〉,〔"〈プロンプト文〉" {; | ,}〕〈変数名〉〔,〈変数名〉...〕

解 説 ・〈待ち時間〉で指定された時間だけ，キーボードからの入力を待ちます．〈待ち時間〉の単位は0.1秒です．

・INPUT WAIT文は，待ち時間に制限があることを除けば，INPUT文と同様です．

・この文の後にマルチステートメントとして他の文を続けた場合，制限時間内に入力がされたときのみ後の文を実行しますが，制限時間内に入力がされなかったときは，後の文を実行せずに次の行へ処理が移ります．プログラミングの際は注意してください．

例 INPUT WAIT 100, "Your name"; NA$

・"Your name?"を表示して，文字列の入力を10秒間待ちます．文字列が入力されると，それを変数NA$に代入します．

プログラム例

```
100 ' INPUT WAIT ｻﾝﾌﾟﾙ
110 *ENTRY
120  A=INT(RND*10):B=INT(RND*10)
130  PRINT A;"+";B"=";
140  INPUT WAIT 30,ANS:GOTO *CALC
150  PRINT:PRINT "時間切れです！",:GOTO *ANS
160 *CALC
170  IF ANS=A+B THEN PRINT "正解です。":GOTO *EXIT
180  PRINT "まちがいです。",
190 *ANS
200  PRINT "答 =";A+B:GOTO *EXIT
210 *EXIT
220  INPUT "続けますか (y/n):";AGAIN$
230  IF AGAIN$="y" THEN *ENTRY ELSE END
Ok
run↵
 2 + 3 =? 5↵
正解です。
続けますか (y/n):? y↵
 3 + 5 =?
時間切れです！              答 = 8
続けますか (y/n):? n↵
Ok
```

・乱数により，たし算の問題を出力し，その答えの入力を3秒間待ちます．

・140行のINPUT WAIT文で，3秒間入力を待ちます．

・3秒以内に入力がなければ，150行の"時間切れです!"を出力し，200行に移って答えを表示します．

・3秒以内に入力があれば，170行で答えを計算し，正しければ210行へ，正しくなければ"まちがいです。"を表示してから，正しい答えを出力します．

INSTR

インストリング：instring

機能　文字列の中から任意の文字列を探し，その文字の位置をバイト数で返す

書式　INSTR (〔〈数式〉,〕〈文字列１〉,〈文字列２〉)

解説　•〈文字列１〉の中から〈文字列２〉を探し，見つかればその位置をバイト数で返します．このとき，〈数式〉は探し始める何バイト目かの位置を指定し，省略した場合は〈文字列１〉の先頭から探し始めます．

•〈文字列２〉が見つからなかったとき，〈文字列１〉がヌルストリングであったとき，〈数式〉が〈文字列１〉の長さより大きいときは０を値として返します．

•〈文字列２〉にヌルストリングを指定すると，〈数式〉と同じ値を返します．

• N_{88}-日本語BASICでは，〈文字列1〉および〈文字列2〉に全角文字を指定することができます．全角文字１文字は２バイト分になりますので，INSTR関数で返される値には注意が必要です．たとえば，

INSTR("漢字", "字")

の場合，返されるバイト数は２でなく３になります．

例　INSTR(TX$, " the ") 返される値 (TX$に含まれる"the"の位置)

プログラム例

```
100 ' INSTR ｻﾝﾌﾟﾙ
110 A$="abcdefghijklmnopqrstuvwxyz"
120 ALP$=INPUT$(1)
130 ALP=ASC(ALP$)
140 IF ALP>64 AND ALP<91 THEN ALP=ALP+32
150 A=INSTR(A$,CHR$(ALP))
160 IF A=0 THEN PRINT ALP$;" はｱﾙﾌｧﾍﾞｯﾄではありません":END
170 PRINT USING "! は##番目のｱﾙﾌｧﾍﾞｯﾄです";ALP$,A
180 END
Ok
run↵
t は20番目のｱﾙﾌｧﾍﾞｯﾄです
Ok
run↵
Z は26番目のｱﾙﾌｧﾍﾞｯﾄです
Ok
```

•キーボードから入力した文字が，アルファベットの何番目にあるかを出力します．

•150行で，入力されたアルファベットが何番目にあるか調べています．この例では小文字で調べているので，入力されたアルファベットが大文字の場合は，140行で大文字から小文字に変換しています．

・実行例では"t"と"Z"を入力しています.

注意： ・〈文字列 1〉に全角文字,〈文字列 2〉に半角文字を指定した場合,指定の仕方によっては,全角文字を 2 分した文字と〈文字列 2〉が一致した場合などに,INSTR関数が思わぬ値を返すことがあります.

参照： **プログラマーズガイド** 第 6 章 「日本語処理」

INT

インテジャー：integer

機能　小数点以下を切り捨てた整数値を返す

書式　INT(〈数式〉)

解説　•〈数式〉の値を超えない最大の整数値(小数点以下を切り捨てた値)を返します.

例　INT(1.23) 返される値 1

INT(—3.21) 返される値 —4

プログラム例

```
100 ' INT ｻﾝﾌﾟﾙ
110 FOR A=-2 TO 2 STEP .5
120   PRINT "A =";A,"FIX(A) =";FIX(A),"INT(A) =";INT(A)
130 NEXT A
140 END
Ok
run
A =-2          FIX(A) =-2    INT(A) =-2
A =-1.5        FIX(A) =-1    INT(A) =-2
A =-1          FIX(A) =-1    INT(A) =-1
A =-.5         FIX(A) = 0    INT(A) =-1
A = 0          FIX(A) = 0    INT(A) = 0
A = .5         FIX(A) = 0    INT(A) = 0
A = 1.5        FIX(A) = 1    INT(A) = 1
A = 2          FIX(A) = 2    INT(A) = 2
Ok
```

• いくつかの実数に対して，FIX関数とINT関数を実行した結果を出力します.

• このプログラム例はFIX関数と同じものです.

注意：• この関数はCINT関数と異なり，数値の型変換は行いません.

参照：FIX, CINT

KACNV$

N88-日本語

ケー・エー・シー・エヌ・ブイ・ダラー：**kanji ascii convert**

機能 文字列中の全角文字を半角文字に変換した文字列で返す

書式 KACNV$ (〈文字列〉)

解説

- 〈文字列〉に含まれる全角文字を半角文字に変換します．
- 変換される全角文字は，AKCNV$関数で変換可能な文字です．
- 変換できない全角文字はそのまま返します．
- グラフィックシンボルモードでは，実際に画面に全角文字を表示できませんが，日本語モードと同じ動作を行います．

例 KACNV$("ＮＥＣ日本電気") 返される値 ⇒ NEC日本電気

- 文字列中の全角文字"ＮＥＣ"を半角文字に変換します．
- 漢字を半角文字に変換することはできません．

プログラム例

N88-日本語

```
100 ' KACNV$ ｻﾝﾌﾟﾙ
110 A$="イロハＡＢＣ"
120 PRINT A$
130 PRINT KACNV$(A$)
Ok
run⏎
イロハＡＢＣ
ｲﾛﾊABC
Ok
```

- 110行で"イロハＡＢＣ"を文字列A$に代入します．
- 120行で，全角文字の"イロハＡＢＣ"を全角のまま文字列を表示します．
- 130行で，全角文字の"イロハＡＢＣ"を半角文字に変換した文字列を表示します．

参照： AKCNV$，資料7「半角文字/全角文字コード変換表」

KEY

キー：key

機　能　ファンクションキーの内容を定義する

書　式　KEY〈キー番号〉,〈文字列〉

解　説

• ファンクションキーの内容を定義します．

• 各ファンクションキーには，最大15文字までの文字列(コントロール文字を含む)を定義することができます．

• N_{88}-日本語BASICでは，全角文字を含む文字列を定義することができます．全角文字の1文字は2バイトなので，最大7文字までの文字列を定義できます．

• キーボードから直接入力できない文字は，CHR$関数をつないで使います．

• 〈キー番号〉は1～10で指定します．

• N_{88}-BASICのシステム起動時，ファンクションキーには

f・1・・・load "	f・ 6・・・save "
f・2・・・auto	f・ 7・・・key
f・3・・・go to	f・ 8・・・print
f・4・・・list	f・ 9・・・edit　.C_R
f・5・・・run C_R	f・10・・・cont C_R

が定義されています．

• N_{88}-日本語BASICでは，かな漢字変換のためにファンクションキーが使われますので，表示される内容が次のようになります．システム起動時に定義されている内容はN_{88}-BASICと同じです．

(1)　グラフィックシンボルモード

N_{88}-BASICと同様です．

(2)　日本語モード

◦半角文字入力モード

f・1～f・10・・・ファンクションキーに定義されている文字列．システム起動時は，N_{88}-BASICと同様です．

◦全角文字入力モード

f・1・・・全角	f・ 6・・・単漢字
f・2・・・ローマ字	f・ 7・・・非漢字
f・3・・・変換	f・ 8・・・前候補
f・4・・・無変換	f・ 9・・・コード
f・5・・・重変換	f・10・・・カタカナ

なお，定義はどのモードであっても行うことができます．

例 KEY 1, "width 80" + CHR$ (13)

• ファンクションキー1に, "width 80C_R"という内容を定義します.

プログラム例

```
100 ' KEY ｻﾝﾌﾟﾙ
110 KEY LIST
120 FOR I=1 TO 5
130   READ K$:KEY I,K$
140 NEXT I
150 KEY LIST
160 END
170 DATA "auto","files","gosub","list","run"
Ok
run⏎
load "          save "
auto            key
go to           print
list            edit .CR
run CR          cont CR

auto            save "
files           key
gosub           print
list            edit .CR
run             cont CR
Ok
```

• 現在のファンクションキーの内容と, 定義しなおした後のファンクションキーの内容を表示します.

• 130行のKEY文で, I番目のファンクションキーに, 170行のデータのI番目の文字列を割り当てます.

参照：KEY LIST, CHR$

KEY LIST

キー・リスト：key list

機能 ファンクションキーの内容を表示する

書式 KEY LIST

解説

- 現在，ファンクションキーに定義されている内容を画面上に表示します．
- コントロールコードも表示します．

例 KEY LIST

- ファンクションキーの内容を表示します．

実行例

```
key list⏎

load "          save "
auto            key
go to           print
list            edit .CR
runCR           cont CR
Ok
```

- ファンクションキーの内容を表示します．

参照：KEY

KEY(n) ON/OFF/STOP

キー・オン/オフ/ストップ：key(n) on/off/stop

機能 ファンクションキーによる割り込みの許可，禁止，停止を設定する

書式
1) KEY〔(〈キー番号〉)〕ON
2) KEY〔(〈キー番号〉)〕OFF
3) KEY〔(〈キー番号〉)〕STOP

解説
• ファンクションキーが押されたときに発生する割り込みを許可(ON)するか，禁止(OFF)するか，停止(STOP)するかを設定します．

• 〈キー番号〉とは，1から10までのファンクションキーの番号です．これが省略された場合は，すべてのファンクションキーに対して実行します．

1) 割り込み動作を許可します．この命令実行後は，指定された〈キー番号〉のファンクションキーを押すごとに割り込みを発生し，ON KEY GOSUB文によって定義された処理ルーチンに分岐します．

2) 割り込み動作を禁止します．この命令実行後は，ファンクションキーを押しても処理ルーチンへの分岐は起こりません．

3) 割り込み動作を停止します．この命令実行後は，ファンクションキーを押しても押されたことを覚えているだけで，処理ルーチンへの分岐は起こりません．しかし，その後KEY(n) ON文によって割り込みが許可されると，先ほどのファンクションキーを押したことによって処理ルーチンに分岐します．

例 KEY ON

• すべてのファンクションキーに対して，割り込みを許可します．

プログラム例

```
100 ' KEY(n) ON/OFF/STOP ｻﾝﾌﾟﾙ
110 ON KEY GOSUB *DISPLAYDATE
120 CLS
130 KEY(1) STOP
140 LOCATE 10,10:PRINT TIME$
150 LOCATE 10,11:PRINT "f.1 ｷｰ ｦ ｵｼﾃｸﾀﾞｻｲ｡"
160 KEY(1) ON
170 GOTO 130
180 *DISPLAYDATE
190  KEY(1) OFF
200  LOCATE 10,13:PRINT DATE$
210  LOCATE 10,14:PRINT "ﾅﾆｶ､ ｷｰ ｦ ｵｼﾃｸﾀﾞｻｲ｡"
220  IF INKEY$="" THEN 220
230  CLS:KEY(1) ON
240 RETURN
Ok
```

• 現在の時刻を表示し，[f・1] が押されたときに年月日を表示するプログラムです．

• 110行で [f・1] が押されたら，180行以降のサブルーチンに分岐するよう定義します．

• 140行で，現在の時刻を表示しています．

• [f・1] が押されると180行以降のサブルーチンに分岐し，何かキーが押されるまで年月日を表示します．

参照：ON KEY GOSUB

KILL

ディスク

キル：kill

機能 フロッピーディスク上のファイルを削除する

書式 KILL 〈ファイルディスクリプタ〉

解説

• 〈ファイルディスクリプタ〉で指定したファイルをフロッピーディスクから削除します．

• 削除するファイルはクローズされていなければなりません．もし，オープン状態のファイルを削除しようとすると "File already open" エラーとなります．また，書き込み禁止属性の付いたファイルを削除しようとすると "File write protected" エラーとなります．この場合，SET 文で書き込み禁止属性を解除してからKILLコマンドを実行してください．

• KILLコマンドは，すべてのディスクファイル(プログラムファイル，ランダムファイル，シーケンシャルファイル)に対して使用できます．

• ドライブ番号はファイルの前に "n:" と指定します．省略されたときドライブ1が指定されたとみなされます．

例 KILL "2:demo"

• ドライブ2に入っているフロッピーディスクの "demo" というファイルを削除します．

プログラム例

ディスク

```
100 ' KILL ｻﾝﾌﾟﾙ
110 OPEN "2:data3" FOR OUTPUT AS #1
120  PRINT #1,"KILL statement test"
130 CLOSE
140 FILES 2:PRINT
150 KILL "2:data3"
160 FILES 2
170 END
Ok
run
copy       1    data1      1    data2     1    rdata1    1    rdata2    1
data4      1    addres s   1    Ldata     1    data6     1    data7     1
rdata3     1    rdata4     1    fdata     1    data3     1

copy       1    data1      1    data2     1    rdata1    1    rdata2    1
data4      1    addres s   1    Ldata     1    data6     1    data7     1
rdata3     1    rdata4     1    fdata     1
Ok
```

• ドライブ2に入っているフロッピーディスクに "data3" というファイルを作り，その後そのファイルを削除します．

• 110行～130行で作成したファイルを150行で削除しています．

• ファイルを作成した後と，削除した後にFILESコマンドを行っています．

参照：SET

KLEN

N88-日本語

ケー・レン：**kanji length**

機能 文字列に含まれる文字数を返す

書式 KLEN (〈文字列〉)

解説

• 〈文字列〉に含まれる文字数(バイト数ではない)を返します.

• 文字数は半角文字，全角文字ともに1文字として数えます.

• ヌルストリングが〈文字列〉に指定されたときは0を返します.

• グラフィックシンボルモードでは，実際に画面に全角文字を表示できませんが，日本語モードと同じ動作を行います.

例 KLEN(" NEC日本電気 ") 返される値 7

プログラム例

N88-日本語

```
100 ' KLEN ｻﾝﾌﾟﾙ
110 INPUT "漢字を含む文字列を入力してください";K$
120 PRINT "この文字列の長さは";KLEN(K$);"です。"
130 END
Ok
run⏎
漢字を含む文字列を入力してください? N88-日本語ＢＡＳＩＣ⏎
この文字列の長さは 12 です。
Ok
```

• 110行でK$に文字列を代入します.

• 120行で代入した文字列の文字数を返します.

KPLOAD

N88-日本語

ケー・ピー・ロード：**kanji pattern load**

機能 外字(全角文字)の文字パターンを定義する

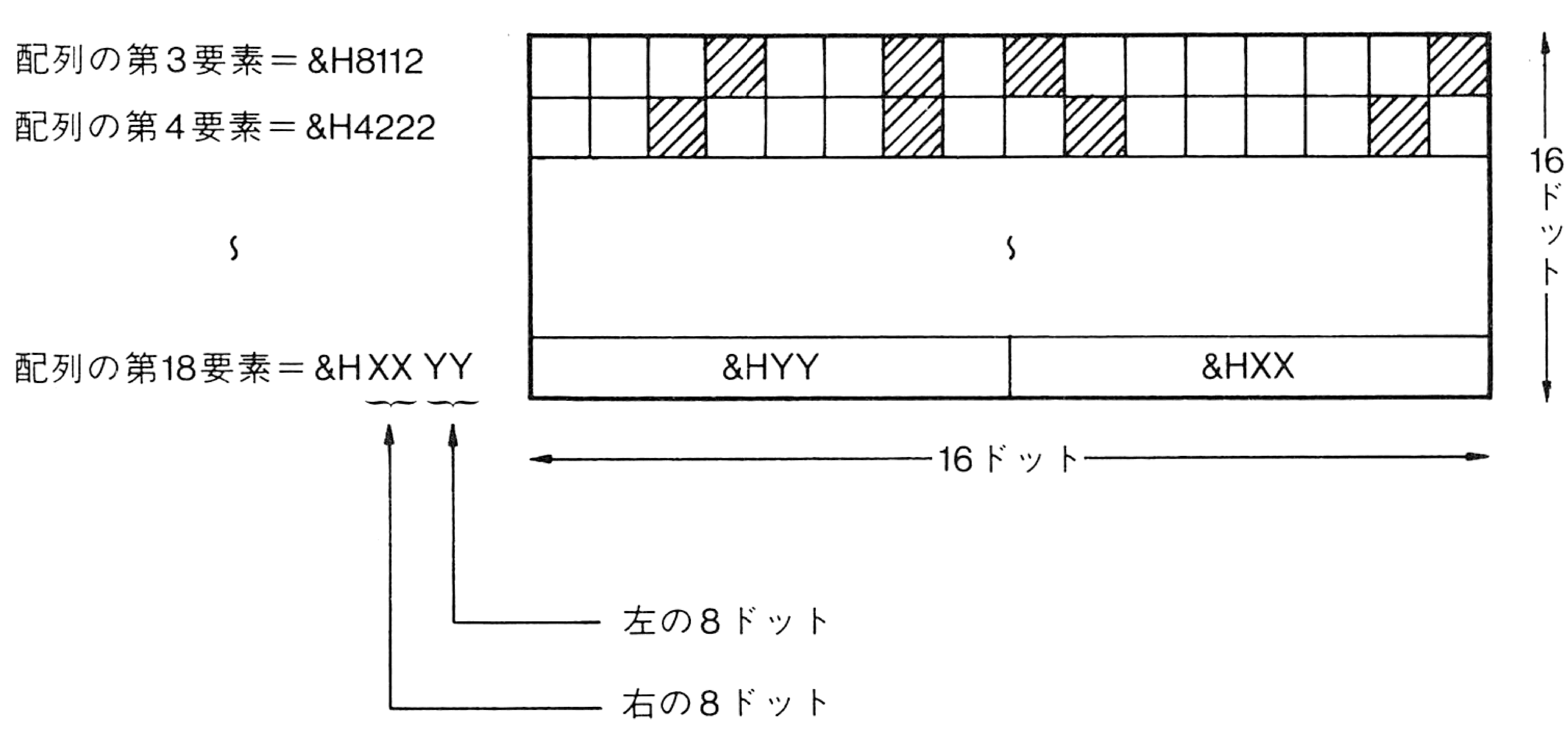
書式 KPLOAD〈文字コード〉,〈整数型配列名〉

解説

- 〈文字コード〉で指定された漢字の文字パターンとして，〈整数型配列名〉で指定されたものを登録します.
- 〈文字コード〉としては，EB9FHからEBDDH(63文字)の値が指定可能です.〈文字コード〉がこの範囲にないときは，"Illegal function call"エラーとなります.
- 〈整数型配列名〉には，一次元の整数型配列を指定します.配列の要素番号を指定することはできません.
- 配列内に格納されている文字パターンは，以下のような形式でなければなりません.

  配列の第1要素＝16

  配列の第2要素＝16
- 配列の第3要素から第18要素までに，以下のように実際の文字パターンのドットイメージを格納してください.



例 KPLOAD &HEB9F , CHRPAT%

- 文字コードEB9FHの外字に，CHRPAT%で指定された外字パターンを定義します.

プログラム例

N88-日本語

```
100 ' KPLOAD ｻﾝﾌﾟﾙ
110 SCREEN 3
120 DIM CHRPTN%(17)
130 CHRPTN%(0)=&HFF
140 CHRPTN%(1)=&HFF
150 CHRPTN%(2)=&HC007     '     *****
160 CHRPTN%(3)=&H3018     '   **      **
170 CHRPTN%(4)=&H821      '  *    *    *
180 CHRPTN%(5)=&H8442     ' *    * *    *
190 CHRPTN%(6)=&H4444     ' *   *   *   *
200 CHRPTN%(7)=&H2288     '*   *     *   *
210 CHRPTN%(8)=&HD297     '*  * ***** *  *
220 CHRPTN%(9)=&H281      '*      *      *
230 CHRPTN%(10)=&HC287    '*    *****    *
240 CHRPTN%(11)=&H2289    '*   *  *  *   *
250 CHRPTN%(12)=&H4285    '*    * * *    *
260 CHRPTN%(13)=&H441     ' *     *     *
270 CHRPTN%(14)=&HE44F    ' *  *******  *
280 CHRPTN%(15)=&H820     '  *         *
290 CHRPTN%(16)=&H3018    '   **      **
300 CHRPTN%(17)=&HE007    '     ******
310 KPLOAD &HEB9F,CHRPTN%
320 PRINT CHR$(&HEB9F)
330 END
Ok
run⏎
```



- 外字を作成して表示します.
- 120行～300行で，外字パターンを配列に定義します.
- 310行で，&HEB9Fの文字コードにパターンを定義します.

注意：• KPLOAD文は，プログラミングによって画面表示用の外字パターンだけを定義するものですが，これによって定義された文字パターンは，プログラムの実行時だけに有効です.

• N88-日本語BASICシステムディスク中の外字辞書に外字を定義するには，"外字管理ユーティリティ"を使います．これによって，表示用の文字パターンとプリンタ印字用の文字パターンの両方を，簡単に定義することができます.

KPOS

N88-日本語

ケー・ポス：**kanji position**

機能 文字列の先頭からの位置をバイト数で返す

書式 KPOS（〈文字列〉,〈数式〉）

解説

- 〈文字列〉の〈数式〉番目の文字が，先頭から何バイト目かを返します．
- 〈数式〉は〈文字列〉の文字を先頭から数えたときの値で半角文字，全角文字ともに1文字として数えていきます．
- 〈数式〉の値は0から255までの範囲とし，これ以外の値を指定した場合は"Illegal function call"エラーとなります．
- 〈文字列〉の文字数が〈数式〉の値に満たない場合は0を返します．
- グラフィックシンボルモードでは，実際に画面に全角文字を表示できませんが，日本語モードと同じ動作を行います．

例 KPOS("NEC日本電気", 6) 返される値 ⇒ 8

- 〈文字列〉"NEC日本電気"の6番目の文字である"電"が，先頭から何バイト目かを返します．
- 〈文字列〉の"NEC"は半角文字（1バイト文字）で，"日本電気"は全角文字（2バイト文字）なので，"電"は8バイト目となります．

プログラム例

N88-日本語

```
100 ' KPOS ｻﾝﾌﾟﾙ
110 CLS
120 A$="ＫＰＯＳのサンプルＬist"
130 PRINT A$
140 LOCATE 1,1:PRINT"ｉ文字目        バイト"
150 FOR I=1 TO 13
160    PRINT I,KPOS(A$,I)
170 NEXT
180 END
Ok
run⏎
ＫＰＯＳのサンプルＬist
 ｉ文字目        バイト
 1               1
 2               3
 3               5
 4               7
 5               9
 6               11
 7               13
 8               15
 9               17
 10              19
 11              21
 12              22
 13              23
Ok
```

• 120行で"KPOSのサンプルList"を変数A$に代入します.

• 150～170行で文字列の1文字目から13文字目までをバイト数で返します.

LEFT$

レフト・ダラー：left $

機能 文字列の左側から指定された長さの文字列を返す

書式 LEFT$ (〈文字列〉,〈数式〉)

解説
- LEFT$関数は〈文字列〉の左側から〈数式〉で指定された長さの文字列を取り出し，それを値として返します．
- 〈数式〉はバイト数を表します．範囲は0～255です．
- 〈数式〉が〈文字列〉の総バイト数以上の場合は〈文字列〉全体を，また〈数式〉が0ならばヌルストリングを返します．

- N88-日本語BASICでは，〈文字列〉に全角文字を指定することができます．全角文字1文字は2バイト分になりますので，LEFT$関数で返される文字列には注意が必要です．たとえば，

  LEFT$("日本語", 2)

の場合，返される文字列は"日本"ではなく"日"となります．

例 LEFT$(TIME$, 2) 返される値 (TIME$変数の左側2バイト)

- カレンダ時計の"時"だけを取り出します．

プログラム例

```
100 ' LEFT$ ｻﾝﾌﾟﾙ
110 INPUT A$
120 FOR I=1 TO 10
130   PRINT LEFT$(A$,I)
140 NEXT I
150 END
Ok
run
? 0123456789
0
01
012
0123
01234
012345
0123456
01234567
012345678
0123456789
Ok
```

- 読み込んだ文字列の左から1バイト目まで，2バイト目まで……10バイト目までの文字を

出力します.

• 130行で，A$の左からIバイト目までの文字を出力します.

注意： •〈文字列〉に全角文字を指定している場合，〈数式〉の値によってはLEFT$関数は全角文字を2分してしまい，思わぬ文字列を返すことがあります.

参照： RIGHT$, MID$，**プログラマーズガイド** 第6章「日本語処理」

L

LEN

レングス：length

機　能　文字列の長さを返す

書　式　LEN（〈文字列〉）

解　説

- 与えられた文字列の長さの値を返します．
- LEN関数は普通の文字の他に，空白や文字として表示されない文字も数えます．

- N_{88}-日本語BASICでは，LEN関数は全角文字１文字を半角文字２文字分として文字列の長さを数えます．たとえば

　　LEN（"日本語"）

では６を返します．

例　LEN(TIME$)　返される値 ⇒　8（＝TIME$変数の長さ）

プログラム例

```
100 ' LEN ｻﾝﾌﾟﾙ
110 INPUT A$:L=LEN(A$)
120 GOSUB 160
130 PRINT "* ";A$;" *"
140 GOSUB 160
150 END
160 ' frame
170 FOR I=1 TO L+4
180   PRINT "*";
190 NEXT I
200 PRINT
210 RETURN
Ok
run↵
? NEC Personal Computer↵
*************************
* NEC Personal Computer *
*************************
Ok
```

- 読み込んだ文字列を"＊"で囲んで出力するプログラムです．
- 110行で，入力した文字列の長さを変数Lに代入しています．
- 160行から210行までは，入力した文字列の長さより"＊"を４つ多くを出力するサブルーチンです．120行と140行で，このサブルーチンを呼び出して枠を作っています．

LET

レット：let

機能 変数に値を代入する

書式 〔LET〕〈変数名〉＝〈式〉

解説

• LETは省略できます.

•〈式〉の内容は数値，文字列のいかなる型であってもかまいませんが，〈変数名〉と〈式〉の値の型が一致しなければなりません．つまり，右辺が数値式のとき左辺は数値変数，右辺が文字式のとき左辺は文字変数でなければなりません.

• 型の異なる数値型変数へ代入する場合は，左辺の型に変換されます.

例 PI＝3.14159

• 数値変数PIに，3.14159という数値を代入します.

プログラム例

```
100 ' LET ｻﾝﾌﾟﾙ
110 LET A=123
120 LET B=.21
130 C=A+B
140 PRINT "A  =";A
150 PRINT "B  =";B
160 PRINT "A+B=";C
170 END
Ok
run⏎
A  = 123
B  = .21
A+B= 123.21
Ok
```

• 変数A, Bにそれぞれ123，0.21を代入し，その和を求めて出力します.

• 130行では，LETを省略した代入文を使っています.

LINE

ライン：line

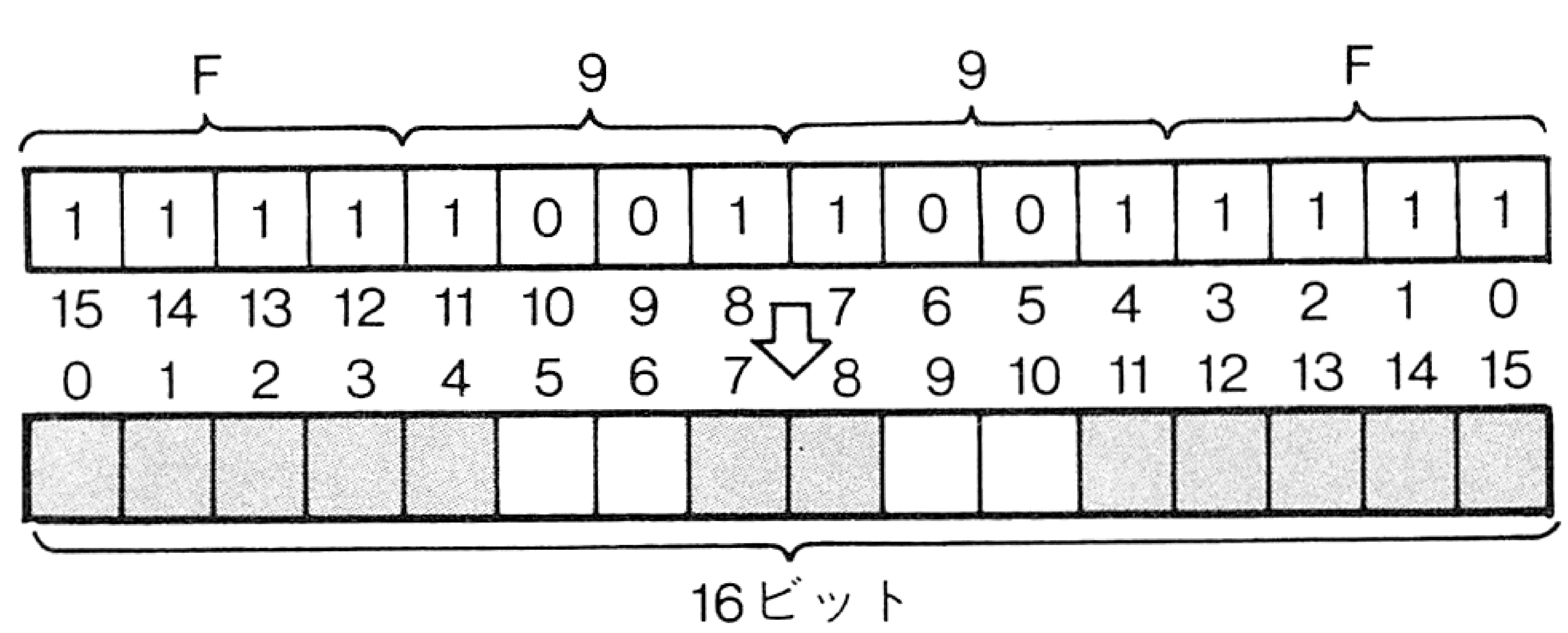
機能 グラフィック画面に直線や長方形を描く

書式 LINE〔{(Wx₁,Wy₁) | STEP(x₁,y₁)}〕−{(Wx₂,Wy₂) | STEP(x₂,y₂)}〔,〈パレット番号〉〕〔,{B | BF}〕〔,〈ラインスタイル〉〕

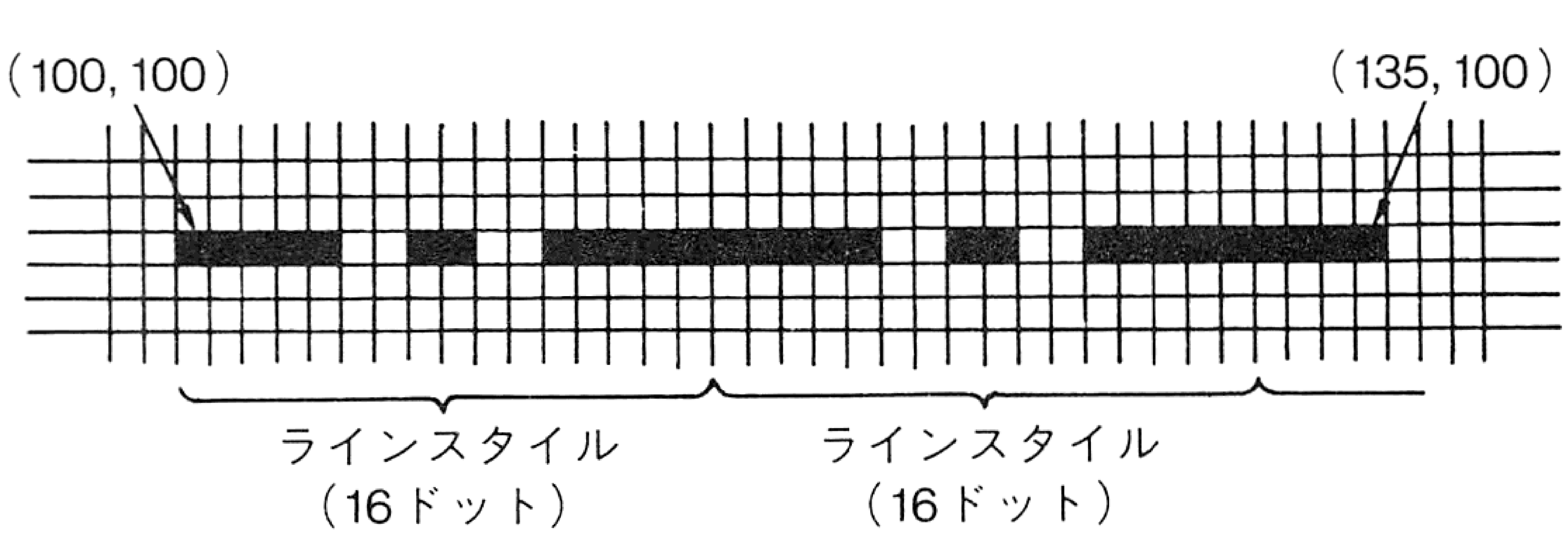
解説

- ワールド座標の2点(Wx_1, Wy_1)と(Wx_2, Wy_2)を結ぶ直線を描きます．(Wx_1, Wy_1)が省略された場合，LPの値をとります．
- 線の色は〈パレット番号〉によって指定します．省略された場合は，COLOR文で指定された〈フォアグラウンドカラー〉が用いられます．
- "B"および"BF"は，指定した2点を対角とする長方形を描きます．"B"はBoxの意味で，枠のみを描きます．"BF"はBox Fillの意味で，その長方形を塗りつぶします．
- 〈ラインスタイル〉は線の形態を設定します．指定できる値は&H0から&HFFFF(16進数)までです．この16進数の数値の16ビットと画面上の16ビットが1対1に対応し，"1"の立っているビットに対応するドットは表示され，"0"のビットに対応するドットは表示されません．線の長さを16ドット以上引いた場合，この〈ラインスタイル〉が画面上の16ドットごとに繰り返されます．次の図は例を実行した場合のものです．水平座標の100から135までの36ドットの間に，1点鎖線の〈ラインスタイル〉が繰り返されています．

ラインスタイル(&HF99F)



※ 同じ番号のビットが対応します．



LINE(100,100)−(135,100), 7, ,&HF99F を実行したとき．

・〈ラインスタイル〉が省略された場合，通常の直線を描きます．また，〈ラインスタイル〉を指定した場合は，BF オプションを指定することはできません．

・LINE 文を実行した場合，LP は(Wx_2, Wy_2)に移動します．

例　LINE(100,100)−(135,100),7,,&HF99F

・ワールド座標(100，100)と(135，100)の2点を結ぶ一点鎖線を，パレット番号7の色で描きます．

プログラム例

```
100 ' LINE ｻﾝﾌﾟﾙ
110 SCREEN 0,0:CLS 3
120 X1=RND*639:Y1=RND*199
130 X2=RND*639:Y2=RND*199
140 CL=RND*6+1:MD=INT(RND*3+1)
150 ON MD GOTO 160,170,180
160 LINE(X1,Y1)-(X2,Y2),CL:GOTO 190
170 LINE(X1,Y1)-(X2,Y2),CL,B:GOTO 190
180 LINE(X1,Y1)-(X2,Y2),CL,BF
190 IF INKEY$="" THEN 120
200 END
Ok
```

・何かキーを押すまで，いろいろな線や四角形を描きます．

・120行から140行で，乱数を使って座標や色を求めています．MD という変数には1から3までの数値が代入されます．

・1が代入されれば160行で直線を，2が代入されれば170行で長方形を，3が代入されれば180行で塗りつぶされた長方形を描きます．

・190行で，何もキーが押されていなければ，120行から実行を繰り返します．

LINE INPUT

ライン・インプット：line input

機能 1行全体(255文字以内)を区切ることなく文字変数に入力する

書式 LINE INPUT〔"〈プロンプト文〉"；〕〈文字変数〉

解説
• 〈プロンプト文〉は，入力データを受ける前に画面に表示される文字列です．

• [↵]の入力までに，キーボードから入力された文字列データが文字変数に代入されます．なお，文字列データを入力せず[↵]だけを押した場合，文字変数はヌルストリングが入力されたとみなされます．

• LINE INPUT 文の実行は，[コントロール] ＋ [C]，または[停止]によって中断することができます．その場合，コマンドレベルに戻り"Ok"と表示します．

例 LINE INPUT "Name？"；NA$

• "Name?"を表示し，文字列の入力を待ちます．文字列が入力されると，それをNA$に代入します．

プログラム例

```
100 ' LINE INPUT ｻﾝﾌﾟﾙ
110 INPUT "文字列をINPUTしてください : ";A$
120 LINE INPUT "文字列をLINE INPUTしてください : ?";B$
130 PRINT:PRINT "INPUTの場合 : ",A$
140 PRINT "LINE INPUTの場合 :",B$
150 END
Ok
run↵
文字列をINPUTしてください : ? ABCD,abcd,1234↵
?Redo from start
文字列をINPUTしてください : ? ABCD↵
文字列をLINE INPUTしてください : ?ABCD,abcd,1234↵

INPUTの場合 :                ABCD
LINE INPUTの場合 :           ABCD,abcd,1234
Ok
```

• INPUT 文とLINE INPUT 文で文字列を入力し，画面に出力します．

• 110行でINPUT 文，120行でLINE INPUT 文を使って文字列を読み込んで，130行と140行で出力しています．

• 実行例では，INPUT 文で1個の変数を指定してあるにもかかわらず，3個のデータが入力されたため，"?Redo from start"というメッセージを表示して，もう一度入力を促しています．

参照：INPUT

LINE INPUT#

ライン・インプット・シャープ：line input #

機能　1行全体(255文字以内)を区切ることなくシーケンシャルファイルから文字変数に代入する

書式　LINE INPUT #〈ファイル番号〉,〈文字変数〉

解説

• 〈ファイル番号〉はOPEN文によって指定したファイル番号です.

• 〈文字変数〉は，行全体が代入される文字変数名です.

• LINE INPUT#文はシーケンシャルファイルからCRコード(⏎が押されたときのコード)までの，すべての文字を区切ることなく読み込みます．読み込める文字数は255文字までです.

• LINE INPUT#文は，データファイルのそれぞれの行がいくつかの欄に分割されているときや，アスキーセーブされたプログラムをデータとして読み込むときなどに有効です.

例　LINE INPUT #1,T$

• ファイル番号1でオープンしたファイルから1行分のデータを読み込み，変数T$に代入します.

プログラム例

ディスク

```
100   LINE INPUT# ｻﾝﾌﾟﾙ
110 OPEN "2:data2" FOR INPUT AS #1
120  IF EOF(1) THEN 160
130  LINE INPUT #1,A$
140  PRINT A$
150  GOTO 120
160 CLOSE
170 END
Ok
run⏎
 In some cases,business computerists become
so incensed as to wage a mental war with s-
oftware,determined to succeed in  spite  of
the documentation. I know of none business-
man who actually refused to look at the do-
cumentation that accompanies  any  software
he acquires. When I asked him why,he smiled
slyly,"It's worthless." Now there  is a man
obsessed.
Ok
```

• ドライブ2に入っているフロッピーディスクの"data2"というファイルの内容を表示します.

• 130行のLINE INPUT#文では，変数A$にCRコード(⏎が押されたときのコード)の前までのすべての文字列が入ります.

参照：OPEN

LINE INPUT WAIT

ライン・インプット・ウェイト：line input wait

機能　キーボードからの入力を時間制限します

書式　LINE INPUT WAIT 〈待ち時間〉, 〔"〈プロンプト文〉" {; | ,}〕〈文字変数〉

解説

- LINE INPUT WAIT 文は，時間制限付きの LINE INPUT 文です．
- LINE INPUT 文と INPUT WAIT 文の両方の機能を兼ね備えています．詳しくはそれぞれの命令を参照してください．

例　LINE INPUT WAIT 50 , "NO＝" , NO $

- "NO＝"を表示して，文字列の入力を5秒間待ちます．文字列が入力されると，それを変数NO$に代入します．

プログラム例

```
100   LINE INPUT WAIT ｻﾝﾌﾟﾙ
110 *ENTRY
120  A$=""
130  FOR I=1 TO 3
140    A$=A$+CHR$(INT(RND*26+65))
150  NEXT I
160  PRINT "問題 : ";A$
170  LINE INPUT WAIT 60,B$:GOTO *ANSWER
180  PRINT:PRINT "時間切れ!!"
190 GOTO *ENTRY
200 *ANSWER
210  IF A$=B$ THEN PRINT "正解!":GOTO *ENTRY
220  PRINT "まちがい!"
230 GOTO *ENTRY
Ok
run⏎
問題 : GHI
GHI⏎
正解!
問題 : NBU
NBV⏎
まちがい!
問題 : MJZ

時間切れ!!
問題 : XSA
[停止]
Break in 170
Ok
```

- タイプ練習プログラムです．
- 乱数により，問題の文字を3文字出力します．
- 問題の文字と同じ文字を入力し⏎を押すと，"正解！"というメッセージが出力されます．

・入力待ち時間は170行で6秒間に設定されています．6秒以内に入力がされない場合，"時間切れ!!"のメッセージが出力されます．

・間違った入力がされた場合には，"まちがい！"のメッセージが出力されます．

・170行で6秒以内に入力された場合には，GOTO＊ANSWERにより，200行へとびます．6秒以内に入力されない場合は，GOTO＊ANSWERは無視されます．

参照：LINE INPUT, INPUT WAIT

LIST/LLIST

リスト/エル・リスト：list/lpt list

機　能　プログラムの内容を出力する

書　式　(1) LIST〔〈行番号〉〕〔－〔〈行番号〉〕〕

(2) LLIST〔〈行番号〉〕〔－〔〈行番号〉〕〕

解　説

• プログラムの内容を画面に表示したり，プリンタにプリントアウトします．LIST コマンドは画面に，LLIST コマンドはプリンタに出力します．

• パラメータを省略すると，プログラムの全部が出力されます．

• 〈行番号〉を１つだけ指定するとその行のみが出力されます．

• ２つの〈行番号〉をハイフン(－)でつないで指定すると，指定した範囲のプログラムが出力されます．このとき，最初の〈行番号〉を省略した場合は，プログラムの先頭からハイフンに続く〈行番号〉までが出力されます．後の〈行番号〉を省略した場合は，指定された〈行番号〉からプログラムの終わりまでが出力されます．

• 〈行番号〉の代わりに，現在の行を表すピリオドも使用できます．

• LIST/LLIST コマンドは，実行し終わるとコマンドレベルに戻ります．

例　LIST －190

• プログラムの先頭から190行までを画面に表示します．

実　行　例

```
    100 ' LIST ｻﾝﾌﾟﾙ
    110 N=1
    120 ' ｽﾀｰﾄ
    130 PRINT N
    140 N=N+1
    150 IF N<10 THEN 130
    160 END
    Ok
(a) list⏎
    100 ' LIST ｻﾝﾌﾟﾙ
    110 N=1
    120 ' ｽﾀｰﾄ
    130 PRINT N
    140 N=N+1
    150 IF N<10 THEN 130
    160 END
(b) list 150⏎
    150 IF N<10 THEN 130
    Ok
(c) list 120-150⏎
    120 ' ｽﾀｰﾄ
    130 PRINT N
    140 N=N+1
    150 IF N<10 THEN 130
    Ok
```

(d)
```
list -140⏎
100 ' LIST ｻﾝﾌﾟﾙ
110 N=1
120 ' ｽﾀｰﾄ
130 PRINT N
140 N=N+1
Ok
```
(e)
```
list 140-⏎
140 N=N+1
150 IF N<10 THEN 130
160 END
Ok
```

(a) プログラムを全部出力します.

(b) 150行だけを出力します.

(c) 120行から150行までを出力します.

(d) プログラムの先頭から140行までを出力します.

(e) 140行からプログラムの終わりまでを出力します.

LOAD

ロード：load

機能 プログラムをメモリにロードする

書式 LOAD 〈ファイルディスクリプタ〉〔,R〕

解説 •〈ファイルディスクリプタ〉で指定されたプログラムファイルをメモリ上にロードします．このとき，以前メモリ上にあったプログラムは消され，変数の値も初期化されます．また，開いているファイルがあった場合は，すべて閉じられます．

•Rオプションを付けると，ファイルを開いたままプログラムをロードし，ただちに実行を開始します．

•LOADコマンドは，指定されたファイルを見つけて実際にプログラムのロードを開始するまでは，メモリ中のプログラムを保存します．

例 LOAD "2:demo"

•ドライブ2に入っているフロッピーディスクから，"demo"というファイル名のついたプログラムをロードします．

実行例

ディスク

```
(a) load "2:test.n88"⏎
    Ok
(b) load "2:test.n88",r⏎
```

•(a)はドライブ2に入っているフロッピーディスクから，"test.n88"というファイルをロードします．

•(b)は，現在開いているファイルを開いたままで，ドライブ2に入っているフロッピーディスクから"test.n88"というファイルをロードして実行を開始します．

参照：SAVE, MERGE, RUN

LOAD?

ロード・クエスチョン：load ?

機能 カセットテープに正しくプログラムがセーブできたかを確認する

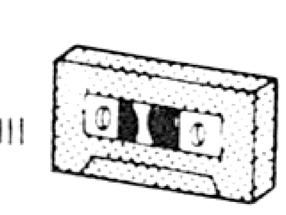
書式 LOAD?〈ファイルディスクリプタ〉

解説
- このコマンドは，カセットファイルに正しくセーブが行われたかを確認する目的で用います.
- このコマンドは，カセットファイルに対してのみ有効であり，他の装置上のファイルが指定された場合には，通常のLOADコマンドと同じ動作をします.
- このコマンドを実行した場合，比較を行うのみで実際のロード動作は行われず，メモリ中のプログラムに変化はありません.
- すべての内容が等しければ"Ok"と表示し，異なれば"Bad"を表示します.

例 LOAD? "CAS:TEST"

- メモリ中のプログラムとカセットファイル上の"TEST"というプログラムを比較し，等しければ"Ok"，異なれば"Bad"を表示します.

実 行 例


カセットテープ

```
save "testpr"
Ok
load? "testpr"
```

- カセットファイルに"testpr"をセーブします.
- load?コマンドで"testpr"が正しくセーブされたかどうかを確認します.
- すべての内容が正しければ"Ok"が表示され，異なっていれば"Bad"を表示します.

LOC

エル・オー・シー：location counter of a file

機能 ファイル中での論理的な現在位置を返す

書式 LOC(〈ファイル番号〉)

解説 ・〈ファイル番号〉で指定されたファイルがランダムファイルであった場合には，そのランダムファイルに対して最後に読み書きされたレコード番号(GET文，PUT文で指定した番号)を返します．

・〈ファイル番号〉で指定されたファイルがシーケンシャルファイルであった場合には，そのファイルがオープンされてから，読み書きされたレコード数を返します．ここで言うレコード数とは，セクタ数をさします．

・指定されたファイルがRS-232Cコミュニケーションファイルであった場合には，LOC関数は割り込みによって受け付けられて，入力バッファ中にたまっているバイト数を返します．

・指定されたファイルがキーボードファイルであった場合には，LOC関数はキーボードからの割り込みによって受け付けられて，キーボードバッファにたまっている文字数を返します．

例 LOC(2) 返される値⇒ (ファイル番号2でオープンされたファイルの現在位置)

プログラム例

ディスク

```
100 ' LOC ｻﾝﾌﾟﾙ
110 OPEN "2:data6" AS #1
120  FIELD #1,20 AS A$,8 AS B$
130  INPUT "名前";NA$:INPUT "給与";KY#
140  LSET A$=NA$:RSET B$=MKD$(KY#):PUT #1
150  PRINT NA$;" さんのデータは";LOC(1);"番です。"
160  INPUT "続けますか(y/n)";C$
170  IF C$="y" THEN PRINT:GOTO 130
180  IF C$<>"n" THEN 160 ELSE CLOSE:END
Ok
run⏎
名前? 寺田⏎
給与? 150000⏎
寺田 さんのデータは 1 番です。
続けますか(y/n)? n⏎
Ok
```

・ランダムファイルを作成しながら，登録したレコード番号を出力します．

・140行で書き込まれたレコード番号を150行で出力しています．この場合，110行でファイル番号1でオープンしているので，LOC関数の引数は1になります．

参照：OPEN, PRINT#, PUT, WRITE#

LOCATE

ロケート：locate

機能 カーソル位置を移動させる

書式 LOCATE〔〈X〉〕〔,〈Y〉〕〔,〈カーソルスイッチ〉〕

解説
- カーソル位置をテキスト画面のキャラクタ座標〈X〉,〈Y〉へ移動します.
- 〈X〉は水平座標を表し，省略した場合は0とみなされます.〈Y〉は垂直座標を表し，省略した場合は現在カーソルが設定されている行とみなされます.
- 〈X〉,〈Y〉は，テキスト画面左上を(0，0)とするキャラクタ座標によって指定します. これらの値をとる範囲は，WIDTH文で指定された〈桁数〉と〈行数〉によって決まります.

  〈X〉＝0〜(〈桁数〉−1)

  〈Y〉＝0〜(〈行数〉−1)

- 〈カーソルスイッチ〉はカーソル状態を決めるスイッチで，1で表示され0で表示されなくなります. キーボードから入力待ちの場合にだけ，〈カーソルスイッチ〉が有効となります.

例

LOCATE 0,10

- キャラクタ座標(0，10)にカーソルを移動させます.

LOCATE 0,0,0

- 画面左上にカーソルを移動させ，さらにカーソルを消します.

プログラム例

- このプログラムは，N88-日本語BASICではグラフィックシンボルモード(SCREEN文の第1パラメータを0, 1, 2のいずれかにする)で実行してください.

```
100 ' LOCATE ｻﾝﾌﾟﾙ
110 CONSOLE 0,25,0,1:WIDTH 80,25:CLS
120 INPUT "x,y";X,Y
130 LOCATE X,Y:PRINT "LOCATE"
140 GOTO 120
Ok
```

- テキスト画面の座標(x, y)を読み込んで，その位置に"LOCATE"を出力します.
- 130行で，xおよびyで指定された位置にカーソルを移動させて"LOCATE"を出力しています.

参照：WIDTH

LOF

エル・オー・エフ：length of a file

機能　ファイルの大きさを返す

書式　LOF(〈ファイル番号〉)

解説

• 〈ファイル番号〉により指定されたファイルがディスクファイルの場合には，LOF関数はそのファイルの大きさをセクタ数で返します．ファイルがランダムファイルの場合には，ファイルに書き込むとき，PUT文で指定した最大のレコード番号に相当します．

• 指定されたファイルがRS-232Cコミュニケーションファイルの場合には，LOF関数はバッファの残りのバイト数を返します．

例　LOF(2) 返される値 (ファイル番号2でオープンしたファイルの大きさ)

プログラム例

ディスク

```
100 ' LOF ｻﾝﾌﾟﾙ
110 OPEN "2:data7" FOR OUTPUT AS #1
120  PRINT #1,123
130 CLOSE #1
140 OPEN "2:data7" FOR INPUT AS #1
150 OPEN "2:data8" AS #2
160  FIELD #2,128 AS A$,128 AS B$
170  PRINT "data7 のセクタ数は";LOF(1);"です。"
180  PRINT "data8 のレコード数は";LOF(2);"です。"
190 CLOSE #1,#2
200 END
Ok
run
data7 のセクタ数は 1 です。
data8 のレコード数は 0 です。
Ok
```

• ドライブ2に入っているフロッピーディスクに"data7"と"data8"というファイルを作り，その大きさを表示します．

• "data7"では120行で123というデータを書き込んでいるため，セクタ数が1になりますが，"data8"ではデータを書き込んでいないためレコード数は0になります．

LOG

ログ：logarithm

機能 自然対数を返す

書式 LOG(〈数式〉)

解説

- 〈数式〉によって与えられた値の自然対数(e=2.71828を底とした対数)を返します.
- 〈数式〉には正の数を指定します.
- 〈数式〉に倍精度実数が含まれる場合は倍精度の値を返しますが，その他の場合には単精度の値を返します.

例

LOG(100) 返される値 ⇒ 4.60517

LOG(1) 返される値 ⇒ 0

LOG(−5) 返される値 ⇒ " Illegal function call "エラー

プログラム例

```
100 ' LOG ｻﾝﾌﾟﾙ
110 INPUT X
120 A=LOG(10)
130 B=LOG(X)
140 PRINT "log10(";X;")=";B/A
150 END
Ok
run⏎
? 100⏎
log10( 100 )= 2
Ok
run⏎
? .12345⏎
log10( .12345 )=-.908509
Ok
```

- 自然対数を用いて常用対数を計算します.
- xが与えられたときの$\log_{10}X$は,

$$\log_{10}(X)=\log_e(X)/\log_e(10)$$

で与えられます.

LPOS

ライン・ポジション：line position

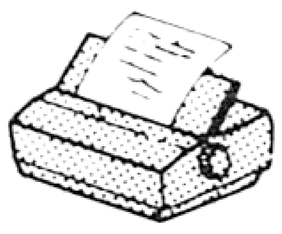
機能 現在のプリンタのヘッド位置を返す

書式 LPOS(〈式〉)

解説
- 現在，プリンタのヘッドがどの位置にあるかを桁数で返します．
- 〈式〉の値は意味を持たず，通常は0を指定します．

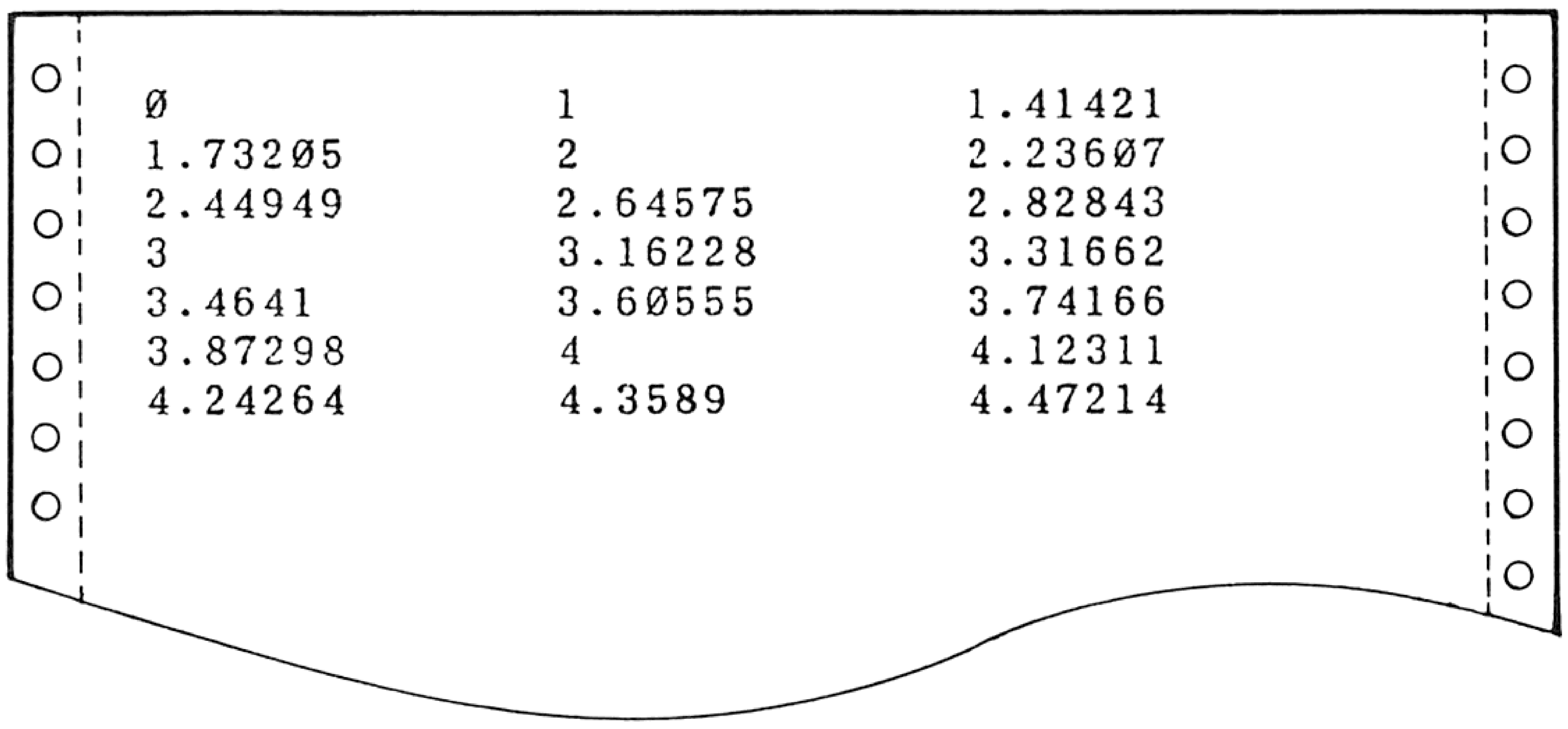
例 LPOS(0) 返される値 (プリンタのヘッド位置)

プログラム例



プリンタ

```
100 ' LPOS ｻﾝﾌﾟﾙ
110 FOR I=0 TO 20
120   IF LPOS(0)>=40 THEN LPRINT
130   LPRINT SQR(I),
140 NEXT I
150 END
Ok
run
Ok
```



```
0           1           1.41421
1.73205     2           2.23607
2.44949     2.64575     2.82843
3           3.16228     3.31662
3.4641      3.60555     3.74166
3.87298     4           4.12311
4.24264     4.3589      4.47214
```

- 0から20までの平方根を，プリンタに出力します．
- 120行でプリンタのヘッドが40桁目を超えた場合は，改行するようにします．

注意：• LPOS関数の値はBASICが管理している値で，プリンタによっては実際のヘッド位置とこの位置とが異なる場合があります．

参照：POS

LSET/RSET

エル・セット/アール・セット：**left set/right set**

機　能　ファイルバッファへデータを書き込む

書　式　LSET 〈文字型変数〉＝〈文字列〉

RSET 〈文字型変数〉＝〈文字列〉

解　説

• FIELD 文で定義した文字型変数に該当するファイルバッファ領域に，データを格納します．

• 格納するデータは文字型データです．数値データは MKI$, MKS$, MKD$ 関数のいずれかにより文字型データに変換してから格納します．

• 〈文字列〉の長さが，FIELD 文で割り当てられた長さより短い場合，LSET 文では左詰め，RSET 文では右詰めでフィールドを満たし，余った部分には空白（CHR$(&H20)）が代入されます．また〈文字列〉がフィールドより長い場合は，LSET 文，RSET 文とも右側の部分が無視されます．

• MKI$, MKS$, MKD$ 関数によって返された〈文字列〉の長さが，FIELD 文で定義された文字型変数の長さより短い場合は，LSET 文を使います．

• 〈文字型変数〉は，FIELD 文であらかじめ定義されていなければなりません．定義されていない変数を用いるとエラーとなります．

例　LSET A$ = " BASIC "

• FIELD 文でファイルバッファに割り当てられた文字型変数 A$ の領域に，"BASIC" というデータを左詰めで書き込みます．

プログラム例

ディスク

```
100 ' LSET/RSET ｻﾝﾌﾟﾙ
110 OPEN "LRdata" AS #1
120  FIELD #1,10 AS A$,10 AS B$
130  INPUT C$
140  LSET A$=C$
150  RSET B$=C$
160  PUT #1,1
170  GET #1,1
180  PRINT A$:PRINT B$
190 CLOSE
200 END
Ok
run↵
? ABC↵
ABC
       ABC
Ok
```

• 文字列を読み込み，それを左詰めと右詰めで書き込みます．

・140行で左詰め，150行で右詰めで読み込んだデータを格納しています.

参照：FIELD, PUT

MAP

マップ：map

機　能　スクリーン座標，ワールド座標の相互変換を行う

書　式　MAP(〈数式１〉,〈機能〉)

解　説

- 〈数式〉で指定されるスクリーン座標あるいはワールド座標を，〈機能〉の指定に従い対応するワールド座標，あるいはスクリーン座標に変換します．
- 〈機能〉は変換の種類を指定し，0から3までの値をとり，それぞれ次の意味を持ちます．

　0：〈数式〉をワールド座標系におけるx座標とみなし，これを対応するスクリーン座標系でのx座標に変換します．(Wx→Sx)

　1：〈数式〉をワールド座標系におけるy座標とみなし，これを対応するスクリーン座標系でのy座標に変換します．(Wy→Sy)

　2：〈数式〉をスクリーン座標系におけるx座標とみなし，これを対応するワールド座標系でのx座標に変換します．(Sx→Wx)

　3：〈数式〉をスクリーン座標系におけるy座標とみなし，これを対応するワールド座標系でのy座標に変換します．(Sy→Wy)

例　MAP(100,1)　返される値⇒　(スクリーン座標系のY座標)

- ワールド座標系のY座標100を，スクリーン座標系のY座標に変換します．

プログラム例

```
100 ' MAP ｻﾝﾌﾟﾙ
110 SCREEN 0,0
120 WINDOW(-100,-100)-(100,100)
130 X=25:Y=70
140 FOR I=0 TO 3 STEP 2
150   M=MAP(X,I):PRINT "map(";X;",";I;")=";M
160   M=MAP(Y,I+1):PRINT "map(";Y;",";I+1;")=";M
170 NEXT I
180 END
Ok
run↵
map( 25 , 0 )= 399
map( 70 , 1 )= 169
map( 25 , 2 )=-92.1753
map( 70 , 3 )=-29.6482
Ok
```

- X座標25，Y座標70をワールド座標系のX座標とY座標，スクリーン座標系のX座標とY座標に変換した値を出力します．
- 120行でウィンドウを(−100，−100)から(100，100)に設定します．

• 140行から170行でワールド座標系，およびスクリーン座標系の値に計算します．

注意：• MAP関数は変換の結果がスクリーン座標系，あるいはワールド座標系から外れてもエラーにならないので注意してください．

MERGE

ディスク

マージ：merge

機能 メモリ上にあるプログラムに，フロッピーディスク上のプログラムを結合する

書式 MERGE〈ファイルディスクリプタ〉

解説 • メモリ上のプログラムに，〈ファイルディスクリプタ〉で指定したプログラムをメモリ上で結合します．

• 指定するプログラムファイルは，前もってアスキーセーブしたものでなければなりません．

• ファイル中のプログラムと，メモリ上のプログラムに同一行番号があった場合には，メモリ上の行はファイルの中の行に置き換わります．

• MERGEコマンドは，実行を終了すると必ずコマンドレベルに戻ります．

例 MERGE "2:result.asc"

• ドライブ2に入っている"result.asc"というファイル名のプログラムファイルを，メモリ上にあるプログラムに結合します．

実行例

ディスク

```
merge "2:test3.n88"⏎
Ok
```

• 現在，メモリ上にあるプログラムとドライブ2に入っているフロッピーディスクに，アスキーセーブされている"test3.n88"というプログラムを結合します．

参照：LOAD, SAVE

MID$

ミッド・ダラー：middle $

機能 文字列の一部を指定された文字列で置き換える

書式 MID$(〈文字変数〉,〈式1〉〔,〈式2〉〕)＝〈文字列〉

解説

・〈文字変数〉の〈式1〉バイト目の位置から〈式2〉バイトの文字を，〈文字列〉の最初から〈式2〉バイトの文字列で置き換えます．

・〈式1〉の値は0より大きく，〈文字変数〉の長さ以下でなければなりません．

・〈式2〉が省略された場合，〈式2〉の値を〈文字列〉のバイト数より多く指定した場合，〈式2〉の値が〈式1〉番目から右のバイト数(残りのバイト数)より大きい場合，〈式2〉の値は(〈文字変数〉の長さ)－(〈式1〉－1)とみなされます．

・〈文字変数〉にヌルストリングを指定することはできません．

・N_{88}-日本語BASICでは，〈文字変数〉や〈文字列〉に全角文字を含むことができます．全角文字1文字は2バイト分になりますので，〈式1〉および〈式2〉の指定の仕方には注意が必要です．たとえば，"最大値"の"大"を"小"に置き換える場合，

```
A$="最大値"
MID$(A$,2,1)="小"
```

とするのではなく

```
MID$(A$,3,2)="小"
```

にします．

例

```
A$ = "ABCDEF"
MID$(A$,4,3) = "XYZ"
```

・変数A$の4バイト目から3バイト("DEF")を"XYZ"に置き換えます．

プログラム例

```
100 ' MID$ ｻﾝﾌﾟﾙ
110 DATA 私の名前は、　　　です。
120 DATA 佐藤,高橋,小早川,かおる
130 READ A$
140 FOR I=1 TO 4
150   READ NA$:MID$(A$,13)=NA$
160   PRINT A$
170 NEXT
180 END
Ok
run
私の名前は、佐藤　です。
```

```
私の名前は、高橋　です。
私の名前は、小早川です。
私の名前は、かおるです。
Ok
```

- 120行に用意してある名前を読み込んで，"私の名前は，……です。"という文を出力するプログラムです．
- 150行のMID$文で変数A$の13バイト目の文字以降，つまり"□□□□□□です。"を変数NA$に読み込んだ文字列で，必要なバイト数だけ置き換えています．

注意： • 〈文字変数〉および〈文字列〉に全角文字を指定した場合，〈式1〉，〈式2〉の指定の仕方によっては，MID$文は思わぬ文字列に置き換えてしまうことがあります．

参照：プログラマーズガイド　第6章「日本語処理」

MID$

ミッド・ダラー：middle $

機能 文字列の指定された位置から任意の長さの文字列を返す

書式 MID$(〈文字列〉,〈式1〉〔,〈式2〉〕)

解説 ・〈文字列〉の〈式1〉バイト目から〈式2〉バイトの文字列を取り出し，値として返します．

・〈式1〉は1～255，〈式2〉は0～255の範囲のバイト数で何バイト目かの値をとります．

・〈式2〉を省略したとき，または〈式2〉が〈式1〉バイト目から右のバイト数(つまり，残りのバイト数)より大きくなったとき，〈式1〉バイト目より右の文字列全部が結果として返されます．

・〈式2〉が0のとき，または〈文字列〉全体のバイト数が〈式1〉より小さい場合，MID$関数はヌルストリングを返します．

・N_{88}-日本語BASICでは，〈文字列〉に全角文字を指定することができます．全角文字1文字は2バイト分になりますので，〈式1〉および〈式2〉の指定には注意が必要です．たとえば，"日本語"の"本"を取り出す場合，

MID$("日本語", 2, 1)

とするのではなく

MID$("日本語", 3, 2)

のようにします．

例 MID$(TIME$,4,2) 返される値 ⇒ (分)

・TIME$変数の4バイト目から2バイトの文字列(＝分)を返します．

プログラム例

```
100 ' MID$ ｻﾝﾌﾟﾙ
110 WIDTH 40,25:CLS
120 A$="NEC Computer"+STRING$(10," ")
130 LOCATE 9,10:GOSUB 190
140 PRINT A$
150 B$=LEFT$(A$,1)
160 A$=MID$(A$,2)+B$
170 GOTO 130
180 ' wait routine
190  FOR I=1 TO 100 :NEXT
200 RETURN
Ok
```

• "NEC Computer" という文字を次々と左に流して表示します．

• このプログラムでは，"NEC□Computer□□□□□□□□□□" という文字列を同じ場所で左に回転させながら表示し，流れていくように見せています．

• 150行と160行でA$に入っている文字列を左に回転させます．たとえば，A$に "NEC□Computer□□□□□□□□□□" が入っていたときは，"EC□Computer□□□□□□□□□□N" となります．

• 190行以降のサブルーチンで少しWAITをかけて，文字の流れが見えるようにしています．

注意： •〈文字列〉に全角文字を指定した場合，〈式１〉および〈式２〉の指定の仕方によっては全角文字を２分してしまい，MID$関数は思わぬ文字列を返すことがあります．

参照：プログラマーズガイド　第６章「日本語処理」

MKI$/MKS$/MKD$

エム・ケー・アイ・ダラー：make integer $
エム・ケー・エス・ダラー：make single $
エム・ケー・ディ・ダラー：make double $

機能　数値を文字コードに変換して返す

書式　MKI$(〈整数表記〉)

MKS$(〈単精度表記〉)

MKD$(〈倍精度表記〉)

解説　・これらの関数は，数値をランダムファイルバッファに対してLSET/RSET文で書き込む際に使用します．

・数値から文字列への変換は，数値が持つ内部表現（2進数表現）の値をそのままそれに対応する文字コードにすることによって行われます．この逆の動作をする関数として，CVI/CVS/CVD関数が用意されています．

・各関数の機能は次のとおりです．

MKI$－整数値を2文字（2バイト）の文字列に変換します．

MKS$－単精度数値を4文字（4バイト）の文字列に変換します．

MKD$－倍精度数値を8文字（8バイト）の文字列に変換します．

例　MKI$(16961) 返される値 "AB"

・整数値16961（内部表現＝4241H）を，2バイトの文字コード"AB"に変換します．

プログラム例

ディスク

```
100 ' MKI$/MKS$/MKD$ ｻﾝﾌﾟﾙ
110 OPEN "2:rdata" AS #1
120  FIELD #1,2 AS D1$,4 AS D2$,8 AS D3$
130  A%=123:B=8534.13:C#=3.14159265358979#
140  LSET D1$=MKI$(A%)
150  LSET D2$=MKS$(B)
160  LSET D3$=MKD$(C#)
170  PUT #1,1
180 CLOSE
190 END
Ok
```

・130行にあるデータを，"rdata"というランダムファイルに書き込みます．

・140行から160行で整数型を2バイトの文字列，単精度実数型を4バイトの文字列，倍精度実数型を8バイトの文字列に型変換をしています．

参照：CVI/CVS/CVD

MON

モニタ：monitor

機能 機械語モニタに制御を移す

書式 MON

解説
- BASICモードから機械語モニタに制御を移すためのコマンドです．
- 機械語モニタがコマンドレベルに入ると，"h]"というプロンプトを表示し，コマンド待ちの状態になります．
- [コントロール] ＋ [B] でBASICモードに戻ります．
- 機械語モニタの使い方については，**プログラマーズガイド** 第19章「機械語モニタ」を参照してください．

例 MON

- 機械語モニタに制御を移します．

実行例

```
MON↵

h] [コントロール]+[B]
Ok
```

- BASICモードから機械語モニタに制御を移します．"h]"が表示されたあと [コントロール] ＋ [B] を押して，BASICのコマンドレベルに戻ります．

注意： • N_{88}-日本語BASICの日本語モードでこのコマンドを実行すると，グラフィックシンボルモードに切り替わってから(SCREEN 0が自動的に実行されてから)，モニタに制御が移されます．この場合，モニタモードからBASICモードに戻ってもグラフィックシンボルモードのままとなります．

参照：プログラマーズガイド 第19章「機械語モニタ」

MOTOR

モータ：motor

機能 カセットテープレコーダのモータを制御する

書式 MOTOR〔〈スイッチ〉〕

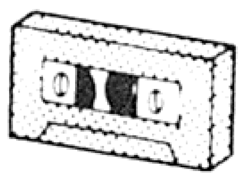
解説

- 〈スイッチ〉を0にすればモータはOFFに，0以外の値にすればONの状態になります．
- 〈スイッチ〉を省略した場合，モータがOFFの状態ならONに，ONの状態ならOFFになります．

例 MOTOR 1

- カセットテープレコーダのモータをONにします．

プログラム例



カセットテープ

```
100 ' MOTOR ｻﾝﾌﾟﾙ
110 MOTOR 1
120 PRINT "テープをセットして、巻きもどして下さい。"
130 INPUT "よろしいですか (y/n) ";A$
140 IF LEFT$(A$,1)<>"y" THEN END
150 PRINT:MOTOR 0
160 PRINT "< PLAY > ボタンを、押して下さい。"
170 INPUT "よろしいですか (y/n) ";A$
180 IF LEFT$(A$,1)<>"y" THEN END
190 MOTOR
200 END
Ok
run↵
テープをセットして、巻きもどして下さい。
よろしいですか (y/n) ? y↵

< PLAY > ボタンを、押して下さい。
よろしいですか (y/n) ? y↵
Ok
```

- 110行でカセットテープレコーダのモータをONの状態にして，カセットテープを巻き戻すことができるようにします．130行で"y"を読み込んだ場合，カセットテープがセットされ，巻き戻されたと判断してひとまずモータをOFFにしておきます．次に，170行で"y"を読み込んだ場合，PLAYボタンが押されていると判断してモータをONにします．

NAME

ネーム：name

機　能　フロッピーディスク上のファイルの名前を変える

書　式　NAME〈旧ファイル名〉AS〈新ファイル名〉

解　説

・〈旧ファイル名〉で表されているフロッピーディスク上のファイルを〈新ファイル名〉に変更します．

・ドライブの指定は，ファイルディスクリプタ中にドライブ番号を含めることによって行います．ドライブ番号を省略した場合，ドライブ1が指定されたとみなされます．

・〈旧ファイル名〉と〈新ファイル名〉のドライブ番号が違う場合，ファイル名の変更は行われません．

・NAMEコマンドによりファイル名の変更を行うとき，そのファイルは閉じられた状態でなければなりません．

例　NAME " sumple " AS " sample "

・ドライブ1に入っているフロッピーディスク上の"sumple"というファイル名を，"sample"に変えます．

プログラム例

```
100 ' NAME ｻﾝﾌﾟﾙ
110 OPEN "2:oldtest" FOR OUTPUT AS #1
120  FOR J=1 TO 10
130    PRINT #1,J
140  NEXT J
150 CLOSE
160 FILES 2:PRINT
170 NAME "2:oldtest" AS "2:newtest"
180 FILES 2
190 END
Ok
run
copy   .    1     data1     1     data2     1     rdata1     1     rdata2     1
data4       1     addres s  1     Ldata     1     data6      1     data7      1
rdata3      1     rdata4    1     fdata     1     oldtes t   1

copy   .    1     data1     1     data2     1     rdata1     1     rdata2     1
data4       1     addres s  1     Ldata     1     data6      1     data7      1
rdata3      1     rdata4    1     fdata     1     newtes t   1
Ok
```

・ドライブ2のフロッピーディスクに"oldtest"というファイルを作成し，"newtest"というファイル名に変更します．

参照：KILL

NEW

ニュー：new

機　能　メモリ上にあるプログラムを消去する

書　式　NEW

解　説

• メモリ上にあるプログラムを消去します．同時に，すべての変数を初期化します．

• NEW コマンドはコマンドレベルにあるとき，新しいプログラムを入力する前に実行します．NEW コマンドの実行が終わるとコマンドレベルに戻ります．

NEW コマンドはオープンされているファイルがある場合，それを自動的に閉じます．

例　NEW

• プログラムを消去します．

プログラム例

```
100 ' NEW ｻﾝﾌﾟﾙ
110 PRINT "NEW コマンドは メモリ上のプログラムを、消去します。"
120 PRINT "このように、プログラム中で使うと、";
130 PRINT "プログラムが消えてしまいます。"
140 NEW
150 END
Ok
run⏎
NEW コマンドは メモリ上のプログラムを、消去します。
このように、プログラム中で使うと、プログラムが消えてしまいます。
Ok
list⏎
Ok
```

• 110行～130行でメッセージを出力して，140行でプログラムを消しています．

NEW CMD

ニュー・シー・エム・ディー：new command

機能 N88-BASIC V2，またはN88-日本語BASICにおいて，拡張命令を使用可能にする

書式 NEW CMD

解説

• NEW CMD文は，N88-BASIC V2，またはN88-日本語BASICで拡張命令を使用できるようにします．

• 拡張命令であるCMD UNLINK文を実行すると，いつでも拡張命令をキャンセルすることができます．

• 拡張命令の追加方法についての詳しい説明は，第3章の拡張命令を参照してください．

例 NEW CMD

• N88-BASIC V2拡張命令を使用可能にします．

実 行 例

```
new cmd⏎
Ok
```

• 拡張命令を使用可能にします．

参照：CMD UNLINK

NEW ON

ニュー・オン：new on

機能 BASICモードからターミナルモードを起動するなど，本機の種々のモードを設定する

書式 NEW ON〈式〉

解説

• 本機を N_{88}-BASICモードにするかターミナルモードにするかなど，最も根本的なモードの選択は，セットアップモードの起動によるメモリスイッチの設定によって決めることができますが，NEW ON文は，それらのモードをソフトウェアによって変える機能があります．

•〈式〉の値は，システムモードスイッチとメモリスイッチが1ビットに対応した16ビット表現の値を指定します．

• NEW ON文で指定できるのは，システムモードスイッチとメモリスイッチのうちの11個です．なお，6，7，14，15ビットは意味を持たないビットです．

• システムモードスイッチ，メモリスイッチとビットとの対応関係を示すと次のようになります．

システムモードスイッチ，メモリスイッチとビットの対応関係

	ビット	
• システムモードスイッチ		
〈BASICモード〉	0	下位
• メモリスイッチ		
〈立ち上げモード〉	1	
〈1行あたりの文字数〉	2	
〈1画面あたりの行数〉	3	
〈Sパラメータ〉	4	
〈DELコード〉	5	
〈パリティチェック〉	8	
〈パリティ〉	9	
〈データビット長〉	10	
〈ストップビット長〉	11	
〈Xパラメータ〉	12	
〈通信方式〉	13	上位

＊6，7，14，15ビットは意味を持ちませんので，0にしてください．

〈BASICモード〉　0：N_{88}-BASIC V1（標準モード）

　　　　　　　　1：N-BASIC[注意：]

〈BASICモード〉　0：BASICモード

　　　　　　　　1：ターミナルモード

〈1行あたりの文字数〉	0: 1行40桁
	1: 1行80桁
〈1画面あたりの行数〉	0: 1画面20行
	1: 1画面25行
〈Sパラメータ〉	0: 無効
	1: 有効
〈DELコード〉	0: DELコードを無視する.
	1: DELコードを処理する.
〈パリティチェック〉	0: パリティ無し
	1: パリティ有り
〈パリティ〉	0: 奇数パリティ
	1: 偶数パリティ
〈データビット長〉	0: 7ビット
	1: 8ビット
〈ストップビット長〉	0: ストップビット1
	1: ストップビット2
〈Xパラメータ〉	0: 無効
	1: 有効
〈通信方式〉	0: 全二重
	1: 半二重

実 行 例

```
new on 2⏎
```

• N_{88}-BASICモードからターミナルモードへ移行します.

注意: • N-BASICで作成されたプログラムを使用するには，本機のシステムモードスイッチをV1Sモードでかつ動作クロックモードスイッチを4MHzに設定し，両面倍密度(2D)のシステムディスクを使ってN_{88}-BASICをスタートしてから，〈BASICモード〉を1に指定してください.

OCT$

オクタル・ダラー：octal $

機能 10進数を8進数の文字列に変換した値を返す

書式 OCT$(〈数式〉)

解説

• 〈数式〉の値を8進数に変換して，その文字列を返します．

• 〈数式〉の値の範囲は，－32768～65535です．〈数式〉の値に小数点以下の値が含まれる場合は，小数点以下を四捨五入したあと変換します．

例

OCT$(－100) 返される値 " 177634 "

OCT$(100) 返される値 " 144 "

OCT$(65535) 返される値 " 177777 "

プログラム例

```
100 '  OCT$ ｻﾝﾌﾟﾙ
110 FOR I=0 TO 16
120   DE$=RIGHT$("   "+STR$(I),3)
130   OC$=RIGHT$("   "+OCT$(I),3)
140   HE$=RIGHT$("   "+HEX$(I),3)
150   PRINT "DECIMAL:";DE$,"OCTAL:";OC$,"HEX:";HE$
160 NEXT I
170 END
Ok
run
DECIMAL:  0   OCTAL:  0    HEX:  0
DECIMAL:  1   OCTAL:  1    HEX:  1
DECIMAL:  2   OCTAL:  2    HEX:  2
DECIMAL:  3   OCTAL:  3    HEX:  3
DECIMAL:  4   OCTAL:  4    HEX:  4
DECIMAL:  5   OCTAL:  5    HEX:  5
DECIMAL:  6   OCTAL:  6    HEX:  6
DECIMAL:  7   OCTAL:  7    HEX:  7
DECIMAL:  8   OCTAL: 10    HEX:  8
DECIMAL:  9   OCTAL: 11    HEX:  9
DECIMAL: 10   OCTAL: 12    HEX:  A
DECIMAL: 11   OCTAL: 13    HEX:  B
DECIMAL: 12   OCTAL: 14    HEX:  C
DECIMAL: 13   OCTAL: 15    HEX:  D
DECIMAL: 14   OCTAL: 16    HEX:  E
DECIMAL: 15   OCTAL: 17    HEX:  F
DECIMAL: 16   OCTAL: 20    HEX: 10
Ok
```

• 0から16までの10進数を，8進数と16進数で表すプログラムです．

• 120行から140行でRIGHT$関数を使っているのは，空白を含め3文字にして桁をそろえて出力するためです．

参照：HEX$

ON COM GOSUB

オン・コム・ゴーサブ：on communication go to subroutine

機能　通信回線からの割り込みが発生したとき，分岐する処理ルーチンの開始行を定義する

書式　ON COM GOSUB〈行番号〉

解説

• 通信回線からRS-232Cのコミュニケーションポートに入力割り込みがあったとき，分岐する処理ルーチンの開始行を定義します．

•〈行番号〉とは，分岐させる処理ルーチンの開始行です．

• 処理ルーチンからの復帰は一般のサブルーチンの場合と同じで，RETURN文によって行います．

• この命令によって割り込み処理ルーチンに分岐させるには，割り込みが発生したとき，COM ONの状態でなければなりません．

• 1つのON COM GOSUB文に，ラベル名と行番号そのものを混在させることはできません．

例　ON COM GOSUB 1000

• コミュニケーションポートに割り込みがあった場合，行番号1000へ分岐するよう定義します．

プログラム例

```
100 ' ON COM GOSUB ｻﾝﾌﾟﾙ
110 CLS
120 OPEN "com1:N81" FOR INPUT AS #1
130  ON COM GOSUB *COMMUNICATE
140  COM STOP
150  IF RIGHT$(TIME$,1)="0" THEN COM ON
160  LOCATE 10,10:PRINT TIME$
170 GOTO 140
180 *COMMUNICATE
190  COM OFF
200  PRINT STRING$(80," ");CHR$(30);
210  A$=INPUT$(1,#1)
220  PRINT A$;
230  IF NOT(EOF(1)) THEN 210 ELES RETURN
Ok
```

• コミュニケーションポートへ入力による割り込みが発生した場合，そのデータを画面に表示します．

• このプログラム例は，COM ON/OFF/STOP文と同じです．

参照：COM ON/OFF/STOP, OPEN, TERM

ON ERROR GOTO

オン・エラー・ゴートゥー：on error goto

機能 エラートラップを可能にし，エラー処理ルーチンの開始行を定義する

書式 ON ERROR GOTO 〈行番号〉

解説

• エラートラップ機能が可能になるとエラーが検出されたとき，指定されたエラー処理ルーチンへプログラムの制御が移ります．

• 〈行番号〉の行が存在しなければ，"Undefined line number"（未定義行）のエラーになります．

• エラートラップ機能を無効にするには，ON ERROR GOTO 0を実行してください．

• エラー処理サブルーチンの実行中にエラーが起きた場合には，それに対応するエラーメッセージが表示され，実行は停止します．エラー処理サブルーチンの中では，エラートラップは起こりません．

• エラー処理後，プログラムの実行を再開するにはRESUME文を使います．

• エラー処理ルーチンにON ERROR GOTO 0の文がある場合には，BASICはエラートラップの原因となったエラーのエラーメッセージを表示し，プログラムの実行を停止します．すべてのエラー処理ルーチンにおいて，エラー回復処理を行わないエラーを検出した場合には，ON ERROR GOTO 0を実行することをお勧めします．

例 ON ERROR GOTO 1000

• エラートラップを可能にします．これを実行した後，エラーが発生したら1000行から実行します．

プログラム例

```
100 ' ON ERROR GOTO ｻﾝﾌﾟﾙ
110 ON ERROR GOTO 160
120 INPUT "x :";X:INPUT "y :";Y
130 IF X=0 AND Y<0 THEN ERROR 250
140 Z=X^Y:PRINT "  y":PRINT " X=";Z
150 GOTO 120
160 IF ERR=250 THEN PRINT " 定義されません。":RESUME 120
170 IF ERR=6   THEN PRINT " オーバーフローです。":RESUME 120
180 ON ERROR GOTO 0
Ok
run↵
x :? 2↵
y :? 10↵
  y
 X= 1024
x :? 100↵
y :? 100↵
 オーバーフローです。
x :? 0↵
y :? -2↵
```

```
定義されません。
X :?[停止]
Break in 12Ø
Ok
```

• 2つの実数XとYを読み込んで，X^Yを出力します．

• 110行でエラーが発生したら，160行から実行するように設定しています．

• Xに0，Yに負の数を入力した場合，130行で250番のエラーを発生させます．そして160行でメッセージを出力したのち120行に戻ります．

• その他のエラーで160行に実行が移ったときは，オーバーフローだけはメッセージを出力し，120行に戻ります．それ以外のエラーはON ERROR 0の文を実行し，原因となったエラーのエラーメッセージを表示してプログラムの実行を停止します．

参照：RESUME, GOTO

ON...GOSUB/ON...GOTO

オン・ゴーサブ/オン・ゴートゥー：on...go to subroutine/on...go to

機能 指定されたいくつかの行に分岐する

書式 ON 〈数式〉 GOSUB 〈行番号〉〔,〈行番号〉...〕

ON 〈数式〉 GOTO 〈行番号〉〔,〈行番号〉...〕

解説 • 〈数式〉の値がプログラムのどの行番号の行に分岐するかを決定します．〈行番号〉の並びは1から始まる数に対応します．たとえば，〈数式〉の値が3であったなら，行番号のリストの3番目の行が分岐点となります．

• 〈数式〉の値の範囲は0〜255です．値が0または行番号リストの個数より大きくなった場合には，次の行へプログラムの制御が移ります．

例 ON N GOTO 100 , 200 , 300

• Nの値によって，100行(N=1)，200行(N=2)，300行(N=3)，次の行(それ以外)へ分岐します．

プログラム例

```
100 ' ON..GOSUB/ON..GOTO ｻﾝﾌﾟﾙ
110 INPUT "N=";N
120 SG=SGN(N)+1
130 ON SG GOTO 150,160
140 PRINT N;"は、マイナスです。":GOTO 110
150 PRINT N;"は、ゼロです。":GOTO 110
160 PRINT N;"は、プラスです。":GOTO 110
Ok
run⏎
N=? 0⏎
 0 は、ゼロです。
N=? 51⏎
 51 は、プラスです。
N=? -24⏎
-24 は、マイナスです。
N=? [停止]
Break in 110
Ok
```

• 単精度実数を読み込んで，その数が正か負かあるいは0かを出力するプログラムです．

• 120行の変数SGの値は，Nが負のときは0，Nが0のときは1，Nが正のときは2になります．130行で，それぞれのメッセージを出力する行に分岐します．

注意：• 1つのON...GOSUB/ON...GOTO文に，ラベルと行番号そのものを混在させることはできません．

参照：GOSUB, GOTO

ON HELP GOSUB

オン・ヘルプ・ゴーサブ：on help go to subroutine

機能 説明 による割り込みルーチンの開始行を定義する

書式 ON HELP GOSUB 〈行番号〉

解説
• 説明 を押したときに，割り込みによって分岐する処理ルーチンの開始行を定義します．
• ON HELP GOSUB 文は，ON KEY GOSUB 文と使い方が同じです．しかし，ファンクションキーの場合は，INPUT 文などキー入力の状態での割り込みは禁止されていますが，説明 の割り込みはそれが可能です．
• 処理ルーチンからの復帰は，一般のサブルーチンの場合と同じで，RETURN 文によって行います．INPUT 文実行中に分岐した後，単にRETURN 文で復帰させると分岐した次の文から実行を再開しますから，行番号を指定するかプログラムでの適切な処理が必要です．
• この命令によって割り込みルーチンに分岐させるには，割り込みがあったとき，HELP ON の状態でなければなりません．

例 ON HELP GOSUB 1000
• 説明 が押されたとき，行番号1000へ分岐するよう定義します．

プログラム例

```
100 ' ON HELP GOSUB ｻﾝﾌﾟﾙ
110 ON HELP GOSUB *MESSAGE
120 HELP ON
130 FOR I=1 TO 3
140   INPUT N
150   PRINT "N =";N,"N*N =";N*N
160 NEXT I
170 HELP OFF:END
180 *MESSAGE
190  HELP OFF
200  PRINT:PRINT "数字を入力してください !!"
210  HELP ON
220 RETURN 140
Ok
run ⏎
? 4 ⏎
N = 4          N*N = 16
? 説明
数字を入力してください !!
? 5 ⏎
N = 5          N*N = 25
? 9 ⏎
N = 9          N*N = 81
Ok
```

• 入力された数値の２乗を出力するプログラムです．

• このプログラム例は，HELP ON/OFF/STOP 文と同じです．

参照：HELP ON/OFF/STOP, ON KEY GOSUB

ON KEY GOSUB

オン・キー・ゴーサブ：on key go to subroutine

機能 ファンクションキーによる割り込みルーチンの開始行を定義する

書式 ON KEY GOSUB 〈行番号〉〔,〈行番号〉...〕

解説
• ファンクションキーを押したときに，割り込みによって分岐する処理ルーチンの開始行を定義します．

• 〈行番号〉とは，割り込みが発生したときに分岐させる処理ルーチンの開始行番号であり，この並びの順序はファンクションキーの番号と1対1に対応しています．

• ファンクションキーは全部で10個ありますから，最大10個の〈行番号〉または〈ラベル名〉を並べて書くことができます．

• 処理ルーチンからの復帰は，一般のサブルーチンと同じで，RETURN文によって行います．

• この命令によって割り込み処理ルーチンに分岐させるには，割り込みが発生したとき，KEY ONの状態でなければなりません．

• 1つのON KEY GOSUB文に，ラベルと行番号そのものを混在させることはできません．

例 ON KEY GOSUB 100,200,300

• [f・1] が押されたら100行へ，[f・2] が押されたら200行へ，[f・3] が押されたら300行へ分岐するように定義します．

プログラム例

```
100 ' ON KEY GOSUB ｻﾝﾌﾟﾙ
110 ON KEY GOSUB *DISPLAYDATE
120 CLS
130 KEY(1) STOP
140 LOCATE 10,10:PRINT TIME$
150 LOCATE 10,11:PRINT "f.1 ｷｰ ｦ ｵｼﾃｸﾀﾞｻｲ｡"
160 KEY(1) ON
170 GOTO 130
180 *DISPLAYDATE
190  KEY(1) OFF
200  LOCATE 10,13:PRINT DATE$
210  LOCATE 10,14:PRINT "ﾅﾆｶ､ ｷｰ ｦ ｵｼﾃｸﾀﾞｻｲ｡"
220  IF INKEY$="" THEN 220
230  CLS:KEY(1) ON
240 RETURN
Ok
```

• 現在の時刻を表示し，[f・1] が押されたときに年月日を表示するプログラムです．

• このプログラム例は，KEY(n) ON/OFF/STOP文と同じものです．

参照：KEY ON/OFF/STOP

ON STOP GOSUB

オン・ストップ・ゴーサブ：on stop go to subroutine

機能 停止 による割り込み処理ルーチンの開始行を定義する

書式 ON STOP GOSUB 〈行番号〉

解説
- 停止 を押したときに，割り込みによって分岐する処理ルーチンの開始行を定義します．
- この命令を実行する場合は，STOP ONの状態でなければなりません．
- 割り込みの対象を 停止 としていることを除けば，ON KEY GOSUB文と同様です．
- この命令を実行すると，本来の 停止 および コントロール ＋ C の機能は失われ，プログラムを中断できなくなります．動作ミスのないプログラムにしてから使用してください．

例 ON STOP GOSUB 100

- 停止 が押されたら，100行へ分岐するように定義します．

プログラム例

```
100 ' ON STOP GOSUB ｻﾝﾌﾟﾙ
110 ON STOP GOSUB *MASK
120 STOP ON
130 F$="###      ##.######^^^^   ##.######^^^^"
140 PRINT "角度";TAB(12);"Sin";TAB(27);"Cos"
150 FOR I=0 TO 180 STEP 20
160   TH=I/180*3.14159
170   PRINT USING F$;I;SIN(TH);COS(TH)
180 NEXT I
190 STOP OFF
200 END
210 *MASK
220 RETURN
Ok
run
角度        Sin             Cos
  0      0.000000E+00    1.000000E+00
 20      3.420200E-01    9.396930E-01
 40      6.427870E-01    7.660450E-01
 60      8.660250E-01    5.000010E-01
 80      9.848080E-01    1.736490E-01
100      9.848080E-01   -1.736470E-01
120      8.660270E-01   -4.999980E-01
140      6.427890E-01   -7.660430E-01
160      3.420230E-01   -9.396920E-01
180      2.808800E-06   -1.000000E+00
Ok
```

- 0から20度ごとに180度までの正弦(SIN)と余弦(COS)を計算します．
- 170行で，PRINT USING文を使って桁をそろえながら，角度とその正弦および余弦を表示します．
- 停止 が押されると210行のサブルーチンへ分岐しますが，RETURN文によってそのま

ま何もせず戻ってきます．つまり，途中でプログラムが中断されないようにしています．

参照：ON KEY GOSUB, STOP ON/OFF/STOP

ON TIME$ GOSUB

オン・タイム・ダラー・ゴーサブ：on time $ go to subroutine

機能 リアルタイムタイマの割り込み発生時刻と，そのとき分岐する処理ルーチンの開始行を定義する

書式 ON TIME$ = "hh : mm : ss" GOSUB 〈行番号〉

解説 • 割り込み時刻の設定の仕方はTIME$変数の場合と同様です．ただし，ここで設定する時刻はリアルタイムタイマに影響を与えません．

• この命令で制御できる分解能は2秒です．場合によっては±1秒程度の誤差が出ることもあります．

• 〈行番号〉は割り込みによって分岐する処理ルーチンの開始行です．

• この命令によって割り込み処理ルーチンに分岐させるには，TIME$ ONの状態でなければなりません．

• 割り込み処理ルーチンから復帰するには，RETURN文を使用します．

例 ON TIME$ = "07 : 30 : 00" GOSUB ＊ MORNINGCALL

• 7時30分0秒に，＊MORNINGCALLというラベル名の処理ルーチンに分岐するよう定義します．

プログラム例

```
100 ' ON TIME$ GOSUB ｻﾝﾌﾟﾙ
110 ON TIME$="06:30:00" GOSUB *MORNING
120 CLS 3
130 TIME$ ON
140 LOCATE 30,10:PRINT TIME$
150 GOTO 140
160 *MORNING
170  TIME$ OFF
180  FOR I=1 TO 10
190    BEEP:PRINT "おはよう !!"
200    FOR J=0 TO 300:NEXT J
210  NEXT I
220 END
Ok
```

• 画面中央部に現在の時刻を表示して，6時30分になったときに，ビープ音とともに"おはよう!!"を10回表示します．

• 110行で，6時30分になったら160行へ分岐するように定義します．

• 140行では，現在の時刻を表示します．

• 6時30分になると，160行以降のサブルーチンでは，ビープ音を鳴らしながら"おはよう!!"を10回表示します．

注意：割り込み時刻は，設定した時間より，わずかですが遅れぎみになります．

参照：TIME$ ON/OFF/STOP, TIME$

OPEN

オープン：open

機能 ファイルをオープンする

書式 OPEN〈ファイルディスクリプタ〉〔FOR〈モード〉〕AS〔#〕〈ファイル番号〉

解説

•〈ファイルディスクリプタ〉で指定されたファイルを，指定された〈ファイル番号〉でオープンします．以後，オープンされたファイルへの入出力は〈ファイル番号〉を指示することにより行います．

•〈モード〉はファイルへのアクセス方法を指定します．モードには，次の4種類があります．

INPUT――――既存のシーケンシャルファイルから入力を行うことを指示する

OUTPUT――――新しくシーケンシャルファイルを作り，出力を行うことを指示する

APPEND――――既存のシーケンシャルファイルの終わりから追加を行うことを指示する

省略したとき――FOR〈モード〉が省略されると，ランダムファイルに対して入出力を行うことを指示する

•INPUT，APPENDモードでは，指定されたファイルが存在しないと，"File not found"エラーとなります．OUTPUTモードでは，常に指定された名前のファイルを新しく作り，同一名のファイルがあった場合，そのファイルは削除されます．ランダムアクセスでファイルが存在しない場合には，新たにファイルが作られます．

•〈ファイル番号〉は1〜15の値を用いることができますが，これは，システム起動時に"How many files(0-15)?"で指定したファイルの数を超えてはいけません．

•OPEN文は，以後の入出力の際に用いるバッファ領域を確保し，ファイルが閉じられるまでの間，そのバッファは指定したファイルへの入出力操作専用に使われます．このバッファはファイル番号と同一の番号で参照されます．

•RS-232Cコミュニケーションファイルをオープンする際には，パリティ，ビット数等を指定する必要があります．これらはTERMコマンドと共通ですので，そちらを参照してください．

•COM1:とCAS1:，CAS2:はハードウェア上の共通ポートを使用するために，同時にオープンすることはできません．

•カセットテープに対してはAPPENDモードの指定はできません．

•RS-232Cコミュニケーションファイルは，〈モードの指定〉を省略すれば送信・受信の両方に使用できます．

例 OPEN "sample" FOR INPUT AS #1

• "sample" というファイル名のシーケンシャルファイルを，入力ファイルとしてファイル番号1で開きます．

プログラム例

ディスク

```
100 ' OPEN ｻﾝﾌﾟﾙ
110 OPEN "2:data9" FOR INPUT AS #1
120 OPEN "2:newdata9" FOR OUTPUT AS #2
130  IF EOF(1) THEN 170
140  INPUT #1,A$
150  PRINT #2,A$
160  GOTO 130
170 CLOSE
180 END
Ok
```

• ドライブ2のフロッピーディスクに入っている"data9"というシーケンシャルファイルの内容を，ドライブ2に入っているフロッピーディスク上の"newdata9"というシーケンシャルファイルにコピーします．

注意：• OPEN "COM:を指定した場合，制御線RTS, DTRが有効になり，受信側に送信可能であることを伝えます．これらの制御線はCLOSE文で無効になります．また，制御線CTS, DSR, DCDはデータ転送中必ず有効でなければなりません．送信の途中でCTSの信号を変化させた場合，その間に送られるデータは保証されません．

• OPENされたファイルは，プログラムを終了するまでに必ずCLOSEするようにしてください．

参照：CLOSE, VARPTR, FIELD

オプション・ベース：option base

OPTION BASE

一般ステートメント

機能 配列の添字の下限を宣言する

書式 OPTION BASE {0 | 1}

解説

• 配列の添字の下限を0または1にすることを宣言します．

• 通常，添字の下限は0ですが，OPTION BASE 1を実行すると下限は1になり，以後，配列の添字に0が用いられると"Subscript out of range"エラーが起こります．

• この宣言はプログラム中で配列変数が宣言，または引用された後に行うことはできません．

• 一度宣言すると，再宣言によって変更することはできません．このような場合には，"Duplicate Definition"エラーになります．

• OPTION BASE文は一度宣言すると，RUNコマンド，あるいはCLEAR文を実行するまで解除，変更することはできません．

例 OPTION BASE

• 配列の添字の下限を1にします．

プログラム例

```
100 ' OPTION BASE ｻﾝﾌﾟﾙ
110 OPTION BASE 0
120 DIM A(1)
130 PRINT "A(0) =";A(0),"A(1) =";A(1)
140 CLEAR
150 OPTION BASE 1
160 DIM A(1)
170 PRINT "A(0) =";A(0),"A(1) =";A(1)
180 END
Ok
run
A(0) = 0         A(1) = 0
A(0) =
Subscript out of range in 170
Ok
```

• 配列の添字の下限を変えながら，配列に格納されているデータを参照します．

• 110行で配列の添字の下限を0に設定しています．

• 150行で配列の添字の下限を1に設定します．170行で配列変数Aの添字に0を指定していますので，エラーになります．

参照：DIM, CLEAR, RUN

OUT

アウト：out

機能 出力ポートに1バイトのデータを送る

書式 OUT 〈I/Oアドレス〉,〈数式〉

解説 •〈I/Oアドレス〉は出力ポートの番号，〈数式〉は出力する1バイトのデータです．

•〈I/Oアドレス〉，〈数式〉はともに0〜255の整数値で指定します．

例 OUT &H54, &H03

•カラーパレット0に3(紫)を指定します(ただし，N88-BASIC V1モードの場合のみ)．54Hポートはカラーパレットを制御するポートです．

プログラム例

•次のプログラム例は，N88-BASIC V2モードで実行してください．

```
100 ' OUT ｻﾝﾌﾟﾙ
110 SCREEN 0,0:CLS 3
120 COLOR=(7,7)
130 CIRCLE(320,100),50,7
140 PAINT(320,100),7,7
150 FOR I=0 TO 511
160   IF I>63 THEN 190
170   OUT &H5B,64:OUT &H5B,I
180   GOTO 210
190   J=I¥64:K=I MOD 64
200   OUT &H5B,64+J:OUT &H5B,K
210   FOR T=0 TO 100:NEXT T
220 NEXT I
230 END
Ok
```

•グラフィック画面の中央に白く塗った円を描き，色を次々と変化させます．

•130行と140行で白い円を描いて，その中を白く塗りつぶします．

•150行から220行で円の色を512色に，次々と変えていきます．

•170行と200行で緑，赤，青のカラーデータを2回に分け，OUT文を使って直接パレットを書き換えています．

注意：•OUT文は十分なハードウェアの知識を持たずに使うと，BASICが正常に動作しなくなることがあります．

参照：INP

〔1〕PAINT

ペイント：paint

機能 指定された境界色で囲まれた領域を，指定された色で塗る

書式 PAINT {(Wx, Wy) | STEP(x, y)} 〔,〈領域色〉〕〔,〈境界色〉〕

解説
- ワールド座標(Wx, Wy)で指定された点を含む〈境界色〉で囲まれた領域を，指定された〈領域色〉で塗りつぶします．
- 〈領域色〉，〈境界色〉はともにパレット番号で指定します．〈領域色〉が省略された場合には，COLOR文で指定された〈フォアグラウンドカラー〉が用いられます．〈境界色〉が省略された場合には〈領域色〉と同じパレット番号の色が用いられます．
- (Wx, Wy)は塗り始める座標を指定します．もしこの点が，すでに指定された境界色と同じ色であった場合には，PAINT文は何の画面操作も行いません．PAINT文はビューポート内でのみ働きますので，ビューポートの境界はPAINT文の動作の境界とみなされます．
- 塗り始めの座標がビューポートの範囲外にある場合は，"Illegal function call"エラーとなります．

例 PAINT(50, 50), 4, 7

- パレット番号7の色で囲まれた領域を，ワールド座標(50, 50)を基点として，パレット番号4の色で塗りつぶします．

プログラム例

```
100 ' PAINT ｻﾝﾌﾟﾙ
110 SCREEN 0,0:CLS 3
120 X=INT(RND*639):Y=INT(RND*199)
130 R=RND*50+1:C=RND*6+1
140 CIRCLE(X,Y),R,C
150 PAINT(X,Y),C
160 GOTO 120
Ok
```

- いろいろな円を描き，その円の中を円を描いた色で塗りつぶします．
- 120行および130行で円の中心座標，半径，色を乱数を使って求めています．

注意：• 複雑な図形を塗りつぶすとき，"Out of memory"エラーとなることがあります．

参照：COLOR, VIEW

P

(2)PAINT

ペイント：paint

機能 指定された境界色で囲まれた領域を，指定されたタイルパターンで埋める

書式 PAINT |(Wx, Wy) / STEP(x, y)| [,〈タイルストリング〉][,〈境界色〉][,〈バックグラウンドストリング〉]

解説

• ワールド座標(Wx, Wy)で指定された点を含む〈境界色〉で囲まれた領域を，指定された模様で埋めます．この動作はタイリングと呼ばれ，模様のついたタイルを領域内に敷きつめたり，中間色で領域内を塗ったような効果を出すことができます．

•〈タイルストリング〉は，タイリングに用いられる基本タイルの模様と大きさを決める文字列です．タイルの大きさは，横方向は8ドット分と決められていますが，縦方向の長さは，〈タイルストリング〉の長さで指定することができます．縦方向がnドットのタイルを指定するためには，白黒モードでnバイト，カラーモードで3＊nバイトの長さを必要とします．

• タイルの模様は，〈タイルストリング〉に対応するキャラクタコードのビットパターン(2進数表現)により表現されます．〈タイルストリング〉の指定の方法は，カラーモードと白黒モードとで異なります．白黒モードでは，1バイトを横8ビットの線に対応させて指定します．キャラクタコードは8ドットのうちどのドットを描き，どのドットを消去するかを決定し，2進数表現で1のビットに対応するドットは描かれ，0に対応するドットは消去されます．〈タイルストリング〉がnバイトの長さであれば，そのストリングの表す模様は，このようにして決定される横8ドットのパターンを縦にnバイトのパターン分だけ並べたものになります．したがって，タイルストリングを構成する文字がすべて同じ文字であったり，1文字だけであった場合には，縦線模様になります．

例) CHR$(&HAA)＋CHR$(&H55)

16進数表現	2進数表現	ドットパターン
&HAA	10101010	●○●○●○●○
&H55	01010101	○●○●○●○●

この例は，2バイトからなる簡単なタイルストリングの例です．このパターンを基本タイルとして指定された領域を埋めれば，細かい市松模様になります．

• カラーモードでも白黒モードと同じように，模様はタイルストリングに対応するビットパターンによって決定されますが，白黒モードとは異なり3バイトで横8ドットが構成されます．ストリング中の文字は先頭から，青，赤，緑のドットパターンを決定していきます．

• カラーモードのとき，ストリングの長さに余りがあった場合には，残りの文字は無視されます．また，3文字に満たない場合には，"Illegal function call"エラーとなります．

例)　CHR$(&H55)＋CHR$(&H33)＋CHR$(&H0F)

16進数表現	2進数表現
&H55	01010101
&H33	00110011
&H0F	00001111
	↓↓↓↓↓↓↓↓
パレット番号──→	01234567

上の例は，3バイトからなるタイルストリングの例で，8×1ドットのラインパターンを作ることができます．この3バイトを並べた状態で各ビットを上記のように"たて"に読むと，左から順に0，1，2……7となります．これが各ドットのパレット番号になり，この例でタイリングを行うと，各ドットごとに異なった色を持つ8ドットのラインで領域が埋められます．また，COLOR＝文，あるいはCMD PAL文によってパレットへのカラーコードの割り付けを変化させると，各ドット単位で色を変えることができます．

• 〈バックグラウンドストリング〉は，タイリングを行う領域にすでに描かれている色，あるいはタイルパターンを指定します．この情報はタイリング実行時の補助的な情報で，〈バックグラウンドストリング〉が指定されていないと，タイリング動作が中断されることがあります．省略された場合には，COLOR文によって指定された〈バックグラウンドカラー〉に対応するストリングが用いられます．

• 〈バックグラウンドストリング〉は，8ドット×1ドット分の長さ(白黒モードで1バイト，カラーモードで3バイト)が利用され，それを超える分は無視されます．

• 〈バックグラウンドストリング〉が表す8ドットのパターンが，〈タイルストリング〉の表すパターン中に3回以上連続して現れる場合は "Illegal function call" エラーとなります．

例

```
PAINT(50,50),"33フ",7
```

• パレット番号7の色で囲まれた領域を，ワールド座標(50，50)を基点として，タイルストリング(CHR$(&H33)＋CHR$(&H33)＋CHR$(&HCC))で埋めます．カラーモードで実行すると，緑と紫の縦縞模様となります．

プログラム例

```
100 ' PAINT ｻﾝﾌﾟﾙ
110 SCREEN 0,0:CLS 3
120 FOR Y=0 TO 3
130   FOR X=0 TO 7
140     LINE(X*80,Y*50)-STEP(80,50),7,B
150     B$=CHR$(RND*255):R$=CHR$(RND*255)
160     G$=CHR$(RND*255)
170     TILE$=B$+R$+G$
180     PAINT(X*80+40,Y*50+25),TILE$,7
190   NEXT X
200 NEXT Y
210 END
Ok
```

• タイルパターンを32個表示させます.

• 150行から160行で，青，赤，緑，それぞれのドットパターンを作っています. RND＊255で，16進数の0HからFFHまでの乱数が得られます.

• 170行で文字型変数TILE$に，150行および160行で作ったタイルストリングを代入しています.

• 140行で白い四角形を描き，180行でその中をタイルパターンで塗りつぶします.

注意：• すでにタイリングされている領域を異なるパターンでタイリングしようとすると，時間がかかったり，メモリが不足して"Out of memory"エラーとなることがあります.

PEEK

ピーク：peek

機能　メモリ上の指定された番地の内容を返す

書式　PEEK（〈番地〉）

解説
- 〈番地〉によって指定されたメモリ番地の内容を 0～255 の値で返します．
- 〈番地〉の範囲は0～65535です．小数点が含まれる場合は小数点以下が四捨五入されます．

例　PEEK（&H9000）　返される値 ⇒　（9000H番地の内容）

プログラム例

```
100 ' PEEK ｻﾝﾌﾟﾙ
110 INPUT "開始アドレス(hex)";SA$
120 SA=VAL("&h"+SA$)
130 INPUT "終了アドレス(hex)";EA$
140 EA=VAL("&h"+EA$)
150 PRINT RIGHT$("000"+HEX$(SA),4);": ";
160 FOR I=SA TO SA+15
170   DA=PEEK(I)
180   PRINT RIGHT$("0"+HEX$(DA),2);" ";
190 NEXT I
200 SA=SA+16:PRINT
210 IF SA<=EA THEN 150
220 END
Ok
run↵
開始アドレス(hex)? 8000↵
終了アドレス(hex)? 8050↵
8000: 00 15 00 64 00 3A 8F E9 20 50 45 45 4B 20 73 61
8010: 6D 70 6C 65 00 34 00 6E 00 85 20 22 73 74 61 72
8020: 74 20 61 64 64 72 65 73 73 28 68 65 78 29 22 3B
8030: 53 41 24 00 47 40 78 00 53 41 F1 FF 94 28 22 26
8040: 68 22 F3 53 41 24 29 00 67 00 82 00 85 20 22 65
8050: 6E 64 20 20 20 61 64 64 72 65 73 73 28 68 65 78
Ok
```

- 指定された範囲のメモリの内容を16進数で表示します．
- 150行では16進数を 4 桁に，180行では 2 桁にそろえて出力するために，RIGHT$関数を使って，それぞれの桁より少ない値のときは左側に 0 を補って出力します．

参照：POKE

POINT

ポイント：point

機能 LP(Last referenced point)を変更する

書式 POINT |(Wx, Wy)|
　　　　　　 |STEP(x, y)|

解説 • BASICは，最後にグラフィック操作の行われた座標を覚えています．この基点のことをLP(Last referenced point)といいます．

• POINT文は，グラフィック操作なしに単にLPを設定する場合に使います．

例 POINT(10, 30)

• LPの値を(10，30)に設定します．

プログラム例

```
100 ' POINT ｻﾝﾌﾟﾙ
110 SCREEN 0,0:CLS 3
120 FOR I=0 TO 35
130   POINT STEP(8,3)
140   GOSUB 170
150 NEXT I
160 END
170 LINE -STEP(-50,50),1
180 LINE -STEP(100,0),2
190 LINE -STEP(-50,-50),3
200 RETURN
Ok
```

• 画面の左上から右下に向けて三角形を描きます．

• 130行でLPを少しずつ右下に下げていきます．

• 170行から190行で三角形を描きます．

参照：(1) POINT

[1]POINT

ポイント：point

機能 LP(Last referenced point)の値を返す

書式 POINT(〈機能〉)

解説 • 最後にグラフィック操作の行われた座標(LP)を，〈機能〉の指定によりスクリーン座標，あるいはワールド座標で返します．

• 〈機能〉は0から3の整数値をとります．それぞれの場合にPOINT関数の返す値は次のようになります．

0：最後にグラフィック操作の行われた点のX座標をワールド座標系上の値に変換して返す．

1：同じくY座標を返す．

2：最後にグラフィック操作の行われた点のX座標をスクリーン座標系上の値に変換して返す．

3：同じくY座標を返す．

例

POINT(0) 返される値⇨ X座標をワールド座標系上の値で返す．

POINT(1) 返される値⇨ Y座標をワールド座標系上の値で返す．

プログラム例

```
100 ' POINT ｻﾝﾌﾟﾙ
110 SCREEN 0,0:CLS 3
120 X=RND*639:Y=RND*199:C=RND*6+1
130 PSET(X,Y),C
140 PRINT "x=";POINT(0);"  y=";POINT(1);
150 PRINT "  color=";POINT(POINT(0),POINT(1))
160 IF INKEY$="" THEN 120 ELSE END
Ok
```

• 何かキーが押されるまで点を描き，その点の座標と色を表示します．

• 120行で，乱数を使って点を描く座標と色を求めて，130行で点を描きます．

• 140行および150行で，POINT関数を使って点を打った座標と色を出力します．

参照：(2) POINT

[2]POINT

ポイント：point

機能 スクリーン座標上の指定された座標にあるドットのパレット番号を返す

書式 POINT(Sx, Sy)

解説
- (Sx, Sy)により指定されたスクリーン座標上に表示されているドットの色をパレット番号で返します.
- グラフィック画面がカラーモードのときはパレット番号が返され，白黒モードのときは，点があるとき1，ないとき0が返されます.
- 指定された座標がビューポートの外にある場合は−1を返します.

例 POINT(100, 100) 返される値⇒ (スクリーン座標(100, 100)にあるドットのパレット番号：グラフィック画面がカラーモードのとき)

プログラム例

```
100 ' POINT ｻﾝﾌﾟﾙ
110 SCREEN 0,0:CLS 3
120 X=RND*639:Y=RND*199:C=RND*6+1
130 PSET(X,Y),C
140 PRINT "x=";POINT(0);"  y=";POINT(1);
150 PRINT "  color=";POINT(POINT(0),POINT(1))
160 IF INKEY$="" THEN 120 ELSE END
Ok
```

- 何かキーが押されるまで点を描き，その点の座標と色を表示します.
- このプログラム例は(1) POINT関数と同じものです.

POKE

ポーク：poke

機能 メモリ上の指定された番地へデータを書き込む

書式 POKE 〈番地〉,〈式〉

解説
- 指定されたメインバンクのメモリ上の番地に1バイトのデータを書き込みます.
- 〈番地〉はデータが書き込まれる番地で，2バイトの整数値(0～65535)の値をとります.
- 〈式〉は書き込まれるデータで，1バイトの整数値(0～255)の値をとります.
- 〈番地〉,〈式〉ともに小数点以下は四捨五入されます.

例 POKE &HD000, &H41
- D000H番地に41Hというデータを書き込みます.

プログラム例

```
100 ' POKE ｻﾝﾌﾟﾙ
110 CLEAR ,&HDFFF
120 ' メモリへ書き込む
130 FOR ADD=&HE000 TO &HE00F
140   POKE ADD,DA:DA=DA+1
150 NEXT ADD
160 '.メモリから読み出す
170 FOR ADD=&HE000 TO &HE00F
180   PRINT RIGHT$(" 0"+HEX$(PEEK(ADD)),4);
190 NEXT ADD
200 END
Ok
run⏎
 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
Ok
```

- 直接メモリに0から15までの値を書き込み，それを読み出して出力します.
- 130行から150行で，メモリに0から15まで値を書き込んでいます.

注意：• この命令はメモリの内容を書き換えてしまうため，不用意に使用するとBASICが使っている作業領域を壊してしまい，誤動作の原因となることがあります．メモリマップの状態をよく理解したうえで使用してください.

参照：CLEAR, PEEK

POS

ポジション：position

機能 カーソルの水平位置を返す

書式 POS(〈式〉)

解説

- テキスト画面上の現在のカーソルの水平位置を返します．
- 〈式〉の値は意味を持ちません．通常は0とします．
- 返される値は，0から，そのときの画面の表示桁数までの整数です．

例 POS(0) 返される値 （カーソルの水平位置）

プログラム例

```
100 ' POS ｻﾝﾌﾟﾙ
110 FOR I=32 TO 95
120   IF POS(0)=>32 THEN PRINT
130   PRINT CHR$(I);" ";
140 NEXT I
150 END
Ok
run
  ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [ ¥ ] ^ _
Ok
```

- キャラクタコードの32から95までの文字を出力します．
- 120行で，32桁を超えたら改行するようにしています．

参照：CSRLIN, LOCATE, WIDTH

PRESET

ピー・リセット：point reset

機能 グラフィック画面上の任意のドットを消去する

書式 PRESET ｛(Wx, Wy) ｜ STEP(x, y)｝〔,〈パレット番号〉〕

解説
• 指定したワールド座標のドットを消去します．通常〈パレット番号〉は省略し，COLOR文によって設定されている〈バックグラウンドカラー〉でドットを描きます．

• STEPを付けた場合は相対座標による指定となります．

• 〈パレット番号〉を指定した場合はPSET文と同じ動作をします．

• PRESET文の実行後，LPは(Wx, Wy)に移動します．

例 PRESET(30, 40)

• ワールド座標(30, 40)のドットを消去します．

プログラム例

```
100 ' PRESET ｻﾝﾌﾟﾙ
110 SCREEN 0,0:COLOR ,1:CLS 3
120 LINE(0,100)-(639,100),6
130 LINE(100,0)-(100,200),6
140 '円と正弦曲線を描く
150 P=3.14159
160 FOR T=0 TO 2*P STEP 2*P/100
170   X=100+100*COS(T):Y=100-50*SIN(T)
180   PRESET(X,Y),0
190   X=100+40*T:PRESET(X,Y),0
200 NEXT T
210 END
Ok
```

• 点で，円と正弦曲線を描きます．

• 110行でグラフィック画面を青にします．

• 120行と130行で座標軸を描きます．原点は(100, 100)になります．

• 150行は変数Pに円周率(π)を代入しています．

• 160行から200行で円と正弦曲線を描きます．

• ここで描く円は中心座標(100, 100)，半径100の円ですから，角度Tとすると，円周上の点(X, Y)は，

$$X=100+100\cos T$$

$$Y=100+50\sin T$$

で得られます．ただし，ワールド座標のy座標は普通の座標と逆で下にいくほど値が大きく

なるので，170行の計算は

$$Y = 100 - 50 \sin T$$

で求めています．

参照：COLOR，PSET

PRINT/LPRINT

プリント/エル・プリント：print/lpt print

機能 数値や文字列を画面やプリンタに出力する

書式 PRINT〔〈式〉...〕

LPRINT〔〈式〉...〕

解説 •指定した〈式〉の値をディスプレイ画面に表示(PRINT)したり，プリンタに出力(LPRINT)したりします．

•〈式〉が数値式の場合は数値を，文字式の場合は文字列を出力します．〈式〉が省略されていると改行のみを行います(キャリッジリターンとラインフィードを出力します)．

•〈式〉の区切り記号としてセミコロン(；)を使うと，直前にプリントしたもののすぐ後に続いて次の数値や文字列を出力します．

•PRINT文で値を出力する領域は，あらかじめ各行が14文字ごとに分割されています．〈式〉の区切り記号としてコンマ(，)を使うと，この領域ごとに出力が行われます．

•〈式〉の最後にセミコロンおよびコンマを付けると改行動作を起こしません．

•変数とダブルクォーテーション(")で囲まれた文字列との区切りに限ってセミコロンを省略できます．

•数値を出力した場合，その後ろには1文字の空白が挿入されます．また，数値の前には符号のための桁を確保します(正の数のとき空白，負の数のとき"−"となります)．

•単精度の数値で，指数形式でなくても6桁以下の桁数で精度に影響を及ぼさず表示できるものは実数形式で表示されます．同様に倍精度の数値は，16桁以下の桁数で精度に影響を及ぼさず表示できるものは実数形式の表示になります．

•表示する数値の長さ，文字列の長さが現在のカーソル位置より後方にとれない(それを表示すると次の行にわたってしまう)場合，改行してそれを表示します．

•"PRINT"というキーワードは，入力時に"?"で代用できます．

例 PRINT "computer"

•"computer"という文字列をディスプレイ画面に表示します．

プログラム例

```
100 ' PRINT/LPRINT ｻﾝﾌﾟﾙ
110 INPUT "名前 ";NA$
120 INPUT "性別 ";SE$
130 INPUT "年齢 ";OD
140 PRINT "名前  :",NA$
150 PRINT "性別  :",SE$
160 PRINT "年齢  :",OD
170 END
Ok
```

```
run⏎
名前 ? ライオン⏎
性別 ? オス⏎
年齢 ? 2⏎
名前  :       ライオン
性別  :       オス
年齢  :        2
Ok
```

• 名前，性別，年齢を読み込んで，それを出力します．

• このプログラム例はINPUT文と同じものです．

PRINT#

プリント・シャープ：print #

機能 シーケンシャルファイルにデータを出力する

書式 PRINT #〈ファイル番号〉,〔〈式〉...〕

解説
• 〈ファイル番号〉は，そのシーケンシャルファイルをオープンしたとき指定した番号です．
• 〈式〉は，ファイルに書き込まれる数値式や文字式です．〈式〉が2つ以上あるときは，その間をセミコロン(；)またはコンマ(，)で区切ります．
• 〈式〉のリストの中の数値式はセミコロンで区切ります．(もし区切り記号にコンマを使うと，数値の間に挿入される余分な空白も書き込まれてしまいます．)
• 数値式と文字式を並べて出力する場合，および文字式の値(文字列)を連続して出力するときは，その間に区切り記号としてコンマを出力させます．そうしないと，文字列を続いて出力したとき，それらのデータは連続した文字列となります．また，文字列自身がデータとしてコンマ，セミコロン，意味のある空白，キャリッジリターン(CHR$(13))，ラインフィード(CHR$(10))などを含む場合は，引用符(CHR$(34))によって囲んで出力します．

例 PRINT #1, CHR$(34)+"width 80, 25"+CHR$(34);",";"page 3";",";
• "width 80, 25" と "page 3" という2つの文字列をシーケンシャルファイルに書き出す例です．
• "width 80, 25"という文字列は，コンマを含んでいますので引用符(CHR$(34))で囲みます．また，文字列の直後にはコンマを出力し，各値を区切っています．

プログラム例

ディスク

```
100 ' PRINT# ｻﾝﾌﾟﾙ
110 OPEN "2:Ldata" FOR OUTPUT AS #1
120  INPUT A$
130  IF A$="end" THEN 160
140  PRINT #1,A$
150  GOTO 120
160 CLOSE
170 END
Ok
run
? BASIC
? FORTRAN
? COBOL
? FORTH
? Prolog
? LOGO
? end
Ok
```

• ドライブ2に入っているフロッピーディスクに"Ldata"というシーケンシャルファイルを作ります．

- 140行で，読み込んだ文字列をファイルに書き込んでいます．
- INPUT#文のプログラム例で，ここで作成したファイルの内容を見ることができます．

参照：PRINT# USING, OPEN, INPUT#

PRINT USING/LPRINT USING

プリント・ユージング/エル・プリント・ユージング：print using/lpt print using

機能 文字列，数値を指定した書式で出力する

書式

PRINT / LPRINT USING〈書式制御文字列〉;〈式〉〔 ; / , 〈式〉...〕〔 ; / , 〕

解説

- 〈書式制御文字列〉によって，〈式〉の出力される領域や書式を決定します．
- 〈書式制御文字列〉で指定した書式制御の数が〈式〉の数より少ないときは，〈書式制御文字列〉が繰り返し使われます．

文字の書式制御

！…与えられた文字変数や文字定数の最初の１文字だけを出力します．

&＜ n 個の□(空白)＞&…与えられた文字変数や文字定数の先頭から(n+2)文字の文字列を出力します．与えられた文字列が(n+2)文字より長い場合は余分な文字は無視され，短い場合には文字列は左詰めに出力され，残った部分には空白が出力されます．

@…文字列の代入に使います．複数個指定した場合，1つの"@"に対して〈式〉中の１つの文字列が代入・出力されます．"@"の数が文字列の個数より多い場合，残りの"@"は無視されます．

N_{88}-日本語BASICでは全角文字１文字は２バイトで表されているため，"！"を使用して出力することはできません．

"………@………"；……，LEFT$("漢字"，2)，……

のように使用してください．

数値の書式制御

#…数値を出力する桁数を指定します．指定した桁数より数値の桁数が小さいときには右詰めで出力されます．

．…小数点の位置を指定します．小数点以下の部分で有効ではない桁には０が出力されます．

＋…〈書式制御文字列〉の最初または最後に付けた場合，数値の符号がそれぞれ前または後に出力されます．２個以上の"＋"を並べた場合には，余分の"＋"は後述の制御文字以外の文字と同じ扱いになります．

－…〈書式制御文字列〉の最後に付けた場合，数値が負の数のときに数値の後ろに"－"が出力されます．前に付けたり，２個以上並べた場合には，後述の制御文字以外の文字と同じ扱いになります．

＊＊…〈書式制御文字列〉の先頭に付けた場合，数値領域の左側に空白部分ができたとき，そこを"＊"で埋めて出力します．この"＊＊"は2桁分の領域を確保します．

¥¥…〈書式制御文字列〉の先頭に付けた場合，出力される数値の直前に"¥"を出力します．"¥¥"は2桁分の領域を確保しますが，このうち1桁分は"¥"の出力領域として使われます．後述の指数形式の書式指定を行った場合には，正しく出力されないことがあります．

＊＊¥…〈書式制御文字列〉の先頭に付けた場合，上記の2つ(＊＊と¥¥)の両方の機能となります．"＊＊¥"は3桁分の領域を確保しますが，このうち1桁分は"¥"の出力領域として使われます．

,…桁数指定の"#"の並びの中に置いた場合，数値の整数部が3桁ごとに","で区切られて出力されます．ただし，"."より右側に置いた場合は，数値の最後に","が出力され，3桁ごとの区切りは行われません．

^^^^…桁数指定の"#"の後に付けた場合，数値は指数形式で出力されます．

…上記の制御文字1文字を単に文字として表示するために使用します．""に続く1文字は常に書式制御機能を持たない文字として出力されます．

制御文字以外の文字

以上の制御文字以外の文字(英数字，カナ，グラフィック記号等)を置いた場合，数値の前や後ろにそのキャラクタが出力されます．

数値の領域を超えた場合

指定した数値領域より数値の桁数が大きい場合，数値の直前に"%"が出力されます．また，指定した数値領域より数値の小数点以下の桁数が大きい場合，指定した小数点領域の次の桁数で四捨五入を行います．たとえば領域を"###.#"としたとき数値に123.45が指定されていた場合，小数点第2位で四捨五入され，123.5になります．したがって，四捨五入された数値が領域より大きくなる原因となった場合も，数値の前に"%"が出力されます．

例　PRINT USING "###.##";A,B

- 数値変数A，Bの値を5桁(小数点以下2桁)で出力します．

プログラム例

```
100 ' PRINT USING/LPRINT USING ｻﾝﾌﾟﾙ
110 PRINT USING "!";"NEC Computer"
120 PRINT USING "&          &";"NEC Computer"
130 PRINT USING "#####";123.456
140 PRINT USING "#####.#";123.456
150 PRINT USING "+#####.#";123.456
160 PRINT USING "#####.#-";-123.456
```

```
170 PRINT USING "**###.#";123.456
180 PRINT USING "¥¥###.#";123.456
190 PRINT USING "**¥##.#";123.456
200 PRINT USING "#####,.#";1234.6
210 PRINT USING "#####,.#^^^^";1234.56
220 PRINT USING "¥¥###._-";123.456
230 PRINT USING "#####";123456!
240 PRINT USING "This is a @.";"pen"
250 END
Ok
run
N
NEC Computer
  123
  123.5
 +123.5
  123.5-
**123.5
 ¥123.5
*¥123.5
 1,234.6
 12345.6E-01
 ¥123.-
%123456
This is a pen.
Ok
```

• 指定した文字列あるいは数値を指定した領域や書式で出力します．

注意：• N88-日本語BASICでは，〈書式制御文字列〉中に半角文字の "@", "¥", "^", "−" と同じコードが含まれる全角文字を使用することはできません．たとえば，"噂"のシフトJISコードは895CHです．下位バイトが"@"のキャラクタコード(5CH)と一致してしまいますから，使用できません．

PRINT # USING

プリント・シャープ・ユージング：print # using

機能 文字列，数値を指定した書式でファイルに出力する

書式 PRINT #〈ファイル番号〉, USING〈書式制御文字列〉;〈式〉〔{;|,}〈式〉...〕〔{;|,}〕

解説

• 〈ファイル番号〉で指定されたファイルに対して，文字列や数値を書式指定して出力します.

• PRINT# USING文は，その対象がファイルであることを除けばPRINT USING文と同じです.

例 PRINT #2, USING "####"; A

• 数値変数Aの値を4桁右詰めでファイルに書き込みます.

プログラム例

ディスク

```
100 ' PRINT# USING ｻﾝﾌﾟﾙ
110 OPEN "stest.dat" FOR OUTPUT AS #1
120  FOR I=0 TO 10
130    PRINT #1,USING "###";I^2
140  NEXT I
150 CLOSE
160 END
Ok
```

• 0から10までの2乗の値を3桁ずつ右詰めでファイルに書き込みます.

参照：PRINT USING, PRINT#, OPEN

PSET

ピー・セット：**point set**

機能 グラフィック画面上のドットを描く

書式 PSET |(Wx, Wy) | 〔,〈パレット番号〉〕
　　　　　　| STEP(x, y) |

解説
- 指定したワールド座標にドットを描きます.
- ドットの色は〈パレット番号〉によって指定し，省略された場合にはCOLOR文の〈フォアグラウンドカラー〉が用いられます.
- PSET文の実行後，LPは(Wx, Wy)に移動します.

例 PSET(30, 40), 5
- ワールド座標(30, 40)にパレット番号5の色でドットを描きます(カラーモードのとき).

プログラム例

```
100 ' PSET ｻﾝﾌﾟﾙ
110 SCREEN 0,0:COLOR ,0,,5:CLS 3
120 LINE(0,100)-(639,100),6
130 LINE(100,0)-(100,200),6
140 '円と正弦曲線を描く
150 P=3.14159
160 FOR T=0 TO 2*P STEP 2*P/100
170   X=100+100*COS(T):Y=100-50*SIN(T)
180   PSET(X,Y)
190   X=100+40*T:PSET(X,Y)
200 NEXT T
210 END
Ok
```

- 点で円と正弦曲線を描きます.
- 詳しい説明は，PRESET文のプログラム例を参照してください.

参照：COLOR, PRESET

PUT

プット：put

機　能　バッファ中のデータをランダムファイルに書き出す

書　式　PUT〔#〕〈ファイル番号〉〔,〈数式〉〕

解　説

•〈ファイル番号〉で指定されたバッファの内容をランダムファイルへ書き込みます．指定されたファイルは，ランダムファイルのモードでオープンされていなければなりません．

•〈数式〉はファイルのレコード番号です．〈数式〉が省略された場合，直前に行われたGET#文，PUT#文で参照されたレコードの次に書き込まれます．なお，書き出すデータはあらかじめFIELD文，LSET/RSET文で準備しておかなければなりません．

例

PUT #3,5

•ファイル番号3でオープンしたランダムファイルの5番目のレコードに，現在バッファに入っているデータを書き込みます．

プログラム例

ディスク

```
100   PUT ｻﾝﾌﾟﾙ
110 OPEN "2:address" AS #1
120  FIELD #1,30 AS A$,20 AS B$,50 AS C$
130  LINE INPUT "名前？ ";NA$
140  IF NA$="end" THEN 200
150  LINE INPUT "電話？ ";TL$
160  LINE INPUT "住所？ ";AR$
170  LSET A$=NA$:LSET B$=TL$:LSET C$=AR$
180  PUT #1
190  GOTO 130
200 CLOSE
210 END
Ok
run
名前？ 相川　房子
電話？ 03-123-4567
住所？ 東京都　日本橋
名前？ 土屋　秀美
電話？ 043-123-4567
住所？ 神奈川県　川崎市
名前？ end
Ok
```

•住所録のファイルを作ります．

•170行で，入力したデータをバッファにセットして，180行でフロッピーディスクに書き込んでいます．

参照：OPEN, FIELD, LSET/RSET, GET

PUT@

プット・アットマーク：put @

機能　グラフィックパターンや漢字を画面に表示する

書式　1) PUT〔@〕(Sx, Sy), 〈配列変数名〉〔〈要素〉〕〔, 〈条件〉〕〔, 〈フォアグラウンドカラー〉, 〈バックグラウンドカラー〉〕

2) PUT〔@〕(Sx, Sy), KANJI(〈漢字コード〉)〔, 〈条件〉〕〔, 〈フォアグラウンドカラー〉, 〈バックグラウンドカラー〉〕

解説　1)・GET@文によって配列に読み込まれたグラフィックパターンを画面上の任意の位置に表示します.

・座標(Sx, Sy)はGET@文の場合と同様で, ワールド座標でなくスクリーン座標で指定します. また, この命令を実行するとLPは(Sx, Sy)に移動されます.

・〈配列変数名〉は表示したいグラフィックパターンが格納されている配列名であり, 〈要素〉は配列内のどこからデータを取り始めるかの指定です. 〈要素〉が省略された場合は, 配列の最初から取り始めます. これらの指定は, GET@文で使ったものと同じものを指定します.

・〈条件〉とは, グラフィックパターンを画面に表示する際, いろいろな機能を指定できるもので, 次下のものが用意されています.

PSET	配列内のグラフィックパターンをそのまま表示します.
PRESET	白黒モードの場合は配列内のパターンをリバースして表示します. カラーモードの場合は各ドットのパレット番号を, 7−(そのドットのパレット番号)として表示します.
OR	配列内のグラフィックパターンと, すでにある画面のグラフィックパターンを1ビット(ドット)ごとにOR(論理和)し, その結果を画面に表示します.
AND	配列内のパターンと画面上のパターンをビットごとにAND(論理積)し, その結果を画面に表示します.
XOR	配列内のパターンと画面上のパターンをビットごとにXOR(排他的論理和)し, その結果を画面に表示します.

※　これらの〈条件〉は画面モードによって演算の対象が違います. 白黒モードの場合, ドットがあるかないか(1ビット)を対象とし, カラーモードの場合, 各パレット番号(0〜7：3ビット)を対象とします.

例)　(パレット3)AND(パレット6)→(パレット2)

もし〈条件〉が省略された場合はXORとみなされます.

・最後の〈フォアグラウンドカラー〉，〈バックグラウンドカラー〉は，白黒モードのときに読み込んだパターンをカラーモードにおいて表示するときにのみ有効なオプションパラメータです．この2つのパラメータは，両方とも指定するか，両方とも指定しないかのどちらかしか許されません．

・〈フォアグラウンドカラー〉は，白黒モードで読み込んだ際に白であったドットに対しての色指定で，これをカラーモードにおいて実行すると任意の色に変えることができます．〈バックグラウンドカラー〉は同様に黒であったドットに対しての色指定です．これらはともに0から7のパレット番号によって指定します．

注意：・GET@文とPUT@文は上記した〈フォアグラウンドカラー〉，〈バックグラウンドカラー〉を指定して白黒モードで読み込んだパターンをカラーモードで表示する用途の他は，原則として同一画面モードで使用するようにしてください．

2)・〈漢字コード〉で指定された漢字または非漢字を画面に表示します．

・N_{88}-BASICでは，使用できる文字は，JIS第1水準の漢字2965字と非漢字約700種です．これらの中から任意の文字を〈漢字コード〉(JISコード)によって指定し，画面上に日本語の文章を作成することができます．

・N_{88}-日本語BASICでは，使用できる文字は，JIS第1水準，第2水準の漢字6535字と非漢字約700種です．
これらの中から任意の文字を〈漢字コード〉(シフトJISコードまたはJISコード)によって指定し，画面上に日本語の文章を作成することができます．

・1)のPUT@文ではユーザがGET@文によって配列に読み込んだパターンを画面に表示しますが，漢字のPUT@文では配列内のデータに当たるものがあらかじめ用意されています．このことを除けば，その使い方および機能は1)の場合とほぼ同様です．ただし，漢字パターンは原則として縦16ドット，横16ドットの大きさと決められています(8×8，16×8のものもある)．また，漢字パターンは白黒モードでGET@した場合と同じ形で格納されていますから，カラーモードにおいて〈フォアグラウンドカラー〉，〈バックグラウンドカラー〉で指定してカラー表示させることも可能です．

・ただし，1つのPUT@文では1つの漢字しか表示することができません．

例

```
PUT@(10,10),G%,PSET
```

・スクリーン座標(10，10)を始点として，配列G%に格納されているグラフィックパターンを表示します．

プログラム例

```
100 ' PUT@ ｻﾝﾌﾟﾙ
110 SCREEN 0,0:CLS 3
120 XD=40:YD=20
130 BYTE=((XD+7)¥8)*YD*3+4
140 FACT=BYTE¥2+1
150 DIM G%(FACT)
160 CIRCLE(XD/2-1,YD/2-1),YD/2,6
170 PAINT(XD/2-1,YD/2-1),6,6
180 GET@(0,0)-STEP(XD-1,YD-1),G%
190 FOR X=0 TO 500 STEP 100
200   PUT@(X,100),G%:PUT@(X,100),G%
210 NEXT X
220 END
Ok
```

- 画面左上に黄色の円を描き，それと同じ円が左から右に移動します．
- このプログラム例はGET@文と同じです．

参照：GET@，資料8「日本語コード」

RANDOMIZE

ランダマイズ：randomize

機能　新しい乱数系列を設定する

書式　RANDOMIZE〔〈式〉〕

解説

- 新しい乱数の種(seed)を与えることによって乱数系列を変更します.
- seedは〈式〉によって－32768～32767の間で与えます.
- 〈式〉を省略した場合は，メッセージが表示されseedの要求をしてきます．このとき入力する値は〈式〉と同じ範囲で指定します.

例　RANDOMIZE

- メッセージとともにseedの要求をしてきます.
- seedの入力により新しい乱数系列を作ります.

プログラム例

```
100 ' RANDOMIZE ｻﾝﾌﾟﾙ
110 GOSUB *RNDSUB
120 CLEAR:GOSUB *RNDSUB
130 RANDOMIZE
140 GOSUB *RNDSUB
150 END
160 *RNDSUB
170  FOR I=0 TO 9
180    PRINT INT(RND*10);
190  NEXT I
200  PRINT
210 RETURN
Ok
run⏎
 2  3  3  5  0  7  4  3  9  9
 2  3  3  5  0  7  4  3  9  9
Random number seed (-32768 to 32767)? 653⏎
 0  2  4  9  6  5  2  7  6  7
Ok
```

- 同じ乱数系列により，乱数を2回表示させ，乱数系列を変更した後，乱数を表示します.
- 130行で，乱数のseedを入力し，乱数系列を変更します.

参照：RND

READ

リード：read

機能 DATA文中に格納されている値を変数に読み込む

書式 READ〈変数名リスト〉

解説 • READ文は，DATA文中に格納されている定数を変数に読み込みます．READ文の変数は数値変数でも文字変数でもかまいませんが，対応するDATA文中の定数の型と一致していなければなりません．

• プログラムの実行中読み出される定数の個数が，READ文で定義されているものより多い場合，"Out of DATA" エラーとなります．

• RESTORE文を実行することにより，読み出すDATA文の位置を行単位で指定できます．

例 READ NM$, AG

• DATA文中に格納されている文字定数，数値定数を変数NM$，AGに読み込みます．

プログラム例

```
100 ' READ ｻﾝﾌﾟﾙ
110 FOR I=0 TO 3
120   READ A:PRINT A;
130 NEXT I
140 READ A$:PRINT A$
150 END
160 DATA 3,2,1,0,発射!
Ok
run⏎
 3  2  1  0 発射!
Ok
```

• "　3　2　1　0　発射！"という文字列を出力するプログラムです．

• 110行から130行で，DATA文から数値データを読み込んで出力し，140行でDATA文から"発射！"という文字列を読み込んで出力します．

参照：DATA, RESTORE

REM

リマーク：remark

機能 プログラム中に注釈を入れる

書式 REM〔<注釈文>〕

〔<注釈文>〕

解説 • REM文は，プログラムを見やすくしたり，コメントを書いたりするために使われます．プログラムの実行には影響を与えません．

• REM文では，キーワード "REM" の代わりにアポストロフィ(')を使うことができます．

• REM文ではコロン(：)も注釈の一部となりますので，コロンで区切って他の文を続けることはできません．

• REM文は，BASICによって実行される文の後にコロンで区切って付け加えることができます．

例 REM ＊＊＊ COMMENT ＊＊＊

• 注釈行を示します．

プログラム例

```
100 ' REM ｻﾝﾌﾟﾙ
110 REM   REM ステートメントによってプログラムの中に
120 REM   コメントが書けます。
130 '     REM は（'）で代用できます。
140 END
Ok
run⏎
Ok
```

• このプログラムはすべて注釈文のため何も実行しません．

• 100行と130行では(')でREM文の代用をしています．

注意：• (')を使った注釈文中に "♥"(キャラクタコードE9H)を含む文字列を入れることはできません．必ずREMを使ってください．同様にN_{88}-日本語BASICでも，文字コードにE9Hを含む文字(たとえば "埼"シフトJISコードは8DE9H)を使うことはできません．

RENUM

リナンバー：renumber

機能 プログラムの行番号を付け替える

書式 RENUM〔〈新行番号〉〕〔,〈旧番号〉〕〔,〈増分〉〕

解説 •〈新行番号〉は，新しく付ける行番号の最初の行番号で，省略時は10とみなされます.

•〈旧行番号〉は，行番号の付け替えを始める現在のプログラムの行番号です．省略したときは，そのプログラムの最初の行番号です.

•〈増分〉は，新しく付ける各行番号の間の増分で，省略したときは10とみなされます.

• RENUMコマンドは，GOSUB文，GOTO文，ON．．．GOSUB文，ON．．．GOTO文およびERL変数などで参照している行番号も新しい行番号に対応して変更します．これらの文が参照している行番号の行がRENUMコマンドを実行する前のプログラム中に存在しない場合には "Undefined line xxxxx in yyyyy" というメッセージが表示されます.
この場合，存在しなかった行番号(xxxxx)はRENUMコマンドによって変更されませんが，その文の行番号(yyyyy)は変更されます.

例 RENUM 1000

• メモリ上にあるプログラムの行番号を付け直します．実行後，行番号は1000, 1010, 1020, ．．．となります.

実行例

```
100 ' RENUM ｻﾝﾌﾟﾙ
110 ' RENUM コマンドは指定した行から
120 ' 行番号を付け替えます。
130 '
140 PRINT " RENUM コマンドは指定した行から"
150 PRINT " 行番号を付け替えます。"
160 END
Ok
renum 200,130,20⏎
Ok
list⏎
100 ' RENUM ｻﾝﾌﾟﾙ
110 ' RENUM コマンドは指定した行から
120 ' 行番号を付け替えます。
200 '
220 PRINT " RENUM コマンドは指定した行から"
240 PRINT " 行番号を付け替えます。"
260 END
Ok
```

• 130行以降の行番号を，200行から20行ごとに付け替えます.

RESTORE

リストア：restore

機能 READ文で読むDATA文の位置を指定する

書式 RESTORE〔〈行番号〉〕

解説
- READ文で読み込みを開始するDATA文の位置を行単位で指定します．
- 〈行番号〉を指定すると，指定された行以降のDATA文の内容から読み始めます．〈行番号〉を省略したときは，プログラム中の最初のDATA文の内容から読み始めます．

例 RESTORE 800

- READ文で読み出されるDATA文の位置を800行に設定します．

プログラム例

```
100 ' RESTORE ｻﾝﾌﾟﾙ
110 READ A,B
120 RESTORE:READ C,D
130 RESTORE 180:READ E$,F$
140 PRINT A,B:PRINT C,D:PRINT E$,F$
150 END
160 DATA 1,2
170 DATA 3,4
180 DATA AA,BB
Ok
run⏎
 1              2
 1              2
AA             BB
Ok
```

- 160行のデータを2回出力し，次に180行のデータを出力するプログラムです．
- 120行のRESTORE文によって，次のREAD文は160行のデータから読むことになります．
- 130行では，170行のDATA文を飛ばして180行のデータを読み込んでいます．

参照：DATA, READ

RESUME

リジュウム：resume

機能 エラー回復処理終了後，プログラムの実行を再開する

書式 RESUME 〔0〕 | NEXT | 〈行番号〉

解説 • RESUME文は，ON ERROR GOTO文によってエラー処理ルーチンに分岐したあと，プログラムの実行を再開する場合に用います．

• プログラムの実行を再開する場所に応じて次のように3つの書式を選ぶことができます．

RESUME または RESUME 0 ……………エラーの原因となった文からプログラムの実行が再開されます．このときエラーとなった原因を取り除かないと，再びエラー処理ルーチンへ戻ります．

RESUME NEXT ……エラーの原因となった文のすぐ次の文から実行が再開されます．

RESUME〈行番号〉…〈行番号〉で指定した行から実行が再開されます．

例 RESUME 1000

• エラー処理ルーチンでエラー回復処理後，1000行から実行を再開します．

プログラム例

```
100 ' RESUME ｻﾝﾌﾟﾙ
110 ON ERROR GOTO 300
120 INPUT "X,Y";X,Y
130 A=X/Y
140 PRINT "X/Y =";A
150 END
300 IF ERR=11 THEN PRINT "** 計算できません **":RESUME 120
310 ON ERROR GOTO 0
Ok
run
X,Y? 100,10
X/Y = 10
Ok
run
X,Y? 35,0
** 計算できません **
X,Y? 35,5
X/Y = 7
Ok
```

• 2つの実数X，Yを読み込んでX/Yを出力します．

・130行の除算でエラーが生じた場合，300行でメッセージを出力して，RESUME 文で120行から実行を再開します．

・その他のエラーは310行のON ERROR 0の文を実行し，原因となったエラーのエラーメッセージを表示してプログラムの実行を停止します．

参照：ON ERROR GOTO

RIGHT$

ライト・ダラー：right $

機 能　文字列の右側から指定された長さの文字列を返す

書 式　RIGHT$(〈文字列〉,〈数式〉)

解 説

• RIGHT$関数は〈文字列〉の右側から〈数式〉で指定された長さの文字列を取り出し，それを値として返します．

•〈数式〉はバイト数をあらわします．範囲は0〜255です．

•〈数式〉が〈文字列〉の総バイト数以上の場合は〈文字列〉全体を，また〈数式〉が0ならばヌルストリングを返します．

• N_{88}-日本語BASICでは，〈文字列〉に全角文字を指定することができます．全角文字1文字は2バイト分になりますので，RIGHT$関数で返される文字列には注意が必要です．たとえば，

RIGHT$("日本語", 2)

の場合，返される文字列は"本語" でなく "語" になります．

例　RIGHT$(TIME$, 2) 返される値 (TIME$変数の右側2バイト)

• カレンダ時計の "秒" だけを取り出します．

プログラム例

```
100 ' RIGHT$ ｻﾝﾌﾟﾙ
110 INPUT A$
120 FOR I=1 TO 10
130   PRINT RIGHT$(A$,I)
140 NEXT I
150 END
Ok
run⏎
? 0123456789⏎
9
89
789
6789
56789
456789
3456789
23456789
123456789
0123456789
Ok
```

• 読み込んだ文字列を，右から1バイト目まで，2バイト目まで……10バイト目までの文字を表示します．

• 130行で，変数A$の右からIバイト目までの文字を出力します．

注意：• 〈文字列〉に全角文字を指定している場合，〈数式〉の値によってRIGHT$関数は全角文字を２分してしまい，思わぬ文字列を返すことがあります．

参照：LEFT$, MID$, **プログラマーズガイド** 第６章 「日本語処理」

RND

ランド：random

機　能　乱数の値を返す

書　式　RND〔(〈数式〉)〕

解　説
• 0以上1未満の乱数値(単精度実数型)を返します.
• 発生する乱数は，RUNコマンドおよびCLEAR文が実行されるごとに同系列を取ります.
• RANDOMIZE文によって，その系列を変えることもできます.
• 発生される乱数は，〈数式〉の値によって次のようになります.

〈数式〉が負の場合——あらかじめ決められた乱数系列の初期値に戻します.

〈数式〉が0の場合——1つ前に発生した乱数の値をとります.

〈数式〉が正の場合——次の乱数を発生します.

• 〈数式〉が省略された場合は，正の値として次の乱数を発生します.

例　RND 返される値 (乱数の値)

プログラム例

```
100 ' RND ｻﾝﾌﾟﾙ
110 DIM SUM(6)
120 FOR I=1 TO 100
130   DA=INT(RND*6+1)
140   SUM(DA)=SUM(DA)+1
150 NEXT I
160 FOR I=1 TO 6
170   PRINT I"-";SUM(I);"%"
180 NEXT I
190 END
Ok
run
 1 - 22 %
 2 - 15 %
 3 - 13 %
 4 - 12 %
 5 - 17 %
 6 - 21 %
Ok
```

• 1から6までの数を乱数で発生させて，それぞれ発生した割合を出力するプログラムです.

• 130行で1から6までの整数を発生させています.

参照：RANDOMIZE

ROLL

ロール：roll

機能 グラフィック画面をスクロールさせる

書式 ROLL 〈ドット数〉

解説
- グラフィック画面を，指定された〈ドット数〉だけ上にスクロールします．
- 〈ドット数〉は，画面の縦のドット数を表します．
- 〈ドット数〉は，N_{88}-BASIC V1では1から197の正の値とします．N_{88}-BASICおよびN_{88}-日本語BASICでは，1から197までの正の値と－197から－1までの負の値を扱うことができ，負の値を指定するとスクロールダウンさせることができます．
- N_{88}-BASIC V1モードでROLL文を使用するには，N_{88}-BASICシステムディスクが必要となります．

例 ROLL 16

- グラフィック画面を上方向へ16ドットスクロールさせます．

プログラム例

- このプログラムはN_{88}-BASICモードで実行してください．

```
100 ' ROLL ｻﾝﾌﾟﾙ
110 SCREEN 1,0:CLS 3
120 FOR I=&H3000 TO &H4F00 STEP &H100
130   FOR J=&H21 TO &H7E
140     KCODE=I+J
150     PUT@(X,168),KANJI(KCODE),PSET
160     X=X+20
170     IF X>623 THEN X=0:ROLL 18
180   NEXT J
190 NEXT I
200 END
Ok
```

- 漢字コード表の順に漢字を1行ずつ表示します．
- 行の最後まで表示されると，その行が18ドット上にスクロールし，次の行を表示します．

注意：• N_{88}-日本語BASICの日本語モードではROLL文が使えません．"Illegal function call" エラーとなります．

RUN

ラン：run

機能 プログラムを実行する

書式 1) RUN〔〈行番号〉〕

2) RUN〈ファイルディスクリプタ〉〔,R〕

解説 •RUNの後に何も指定しない場合は，メモリ上にあるプログラムの先頭から実行を開始します．

•〈行番号〉を指定すると，その行から実行が開始されます．

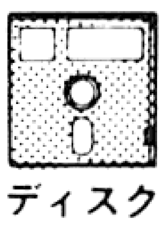
•〈ファイルディスクリプタ〉を指定すると，フロッピーディスクから指定した〈ファイル名〉のプログラムをロードし，実行します．

•RUNコマンドを実行すると，すべての開いているファイルを閉じますが，Rオプションを付けた場合には，すべてのファイルは開いたままになります．

例 RUN "demo"

•ドライブ1のフロッピーディスク上にある"demo"というファイル名のプログラムをロードし，実行します．

実行例



(a) `run`↵

(b) `run 1000`↵

(c) `run"2:test.n88"`↵

(d) `run"2:test.n88",r`↵

•(a)は，現在メモリ上にあるプログラムの先頭から実行を始めます．

•(b)は，現在メモリ上にあるプログラムの1000行から実行を始めます．

•(c)は，ドライブ2のフロッピーディスクに入っている "test.n88" というプログラムをロードして，先頭から実行を始めます．

•(d)は，現在オープンしているファイルを開いたまま，ドライブ2のフロッピーディスクに入っている "test.n88" というプログラムをロードして，先頭から実行を始めます．

SAVE

セーブ：save

機　能　メモリのBASICプログラムをファイルにセーブする

書　式　SAVE〈ファイルディスクリプタ〉〔, {A | P}〕

解　説　•〈ファイル名〉で指定されるファイルにメモリ上のプログラムを書き込みます．指定したファイル名と同じ名前のファイルが存在した場合には，古い内容は失われて新しいものに変更されます．

•Aオプションが指定された場合には，プログラムはアスキー形式でセーブされます．オプションの指定がない場合には，バイナリ形式に圧縮されてプログラムのセーブが行われます．アスキー形式のセーブは，バイナリ形式よりも多くのファイルスペースを必要としますが，MERGEコマンドのように，セーブされたプログラムファイルを操作する場合に用いられます．また，アスキー形式でセーブされたファイルは，データファイルとして読み出すことができます．

•Pオプションが指定された場合には，プログラムは暗号化されたバイナリ型でセーブされます．Pオプションは，セーブされたプログラムを，書き込み，変更といった操作から保護するための機能を持っています．

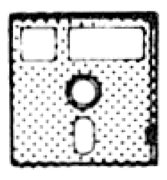
•Pオプションの付いたプログラムは，LISTコマンドやEDITコマンドにより内容を見たり変更しようとすると "Illegal function call" エラーになります．

•Pオプションは，解除することはできません．

例　SAVE "2：sample"

•ドライブ2に入っているフロッピーディスクに，メモリ上にあるプログラムを"sample"というファイル名でセーブします．

実　行　例


ディスク

(a)
```
save "2:test4.n88"⏎
Ok
```
(b)
```
save "2:test3.n88",a⏎
Ok
```

•(a)は，現在メモリ上にあるプログラムを "test4.n88" というファイル名でドライブ2に入っているフロッピーディスクにセーブします．

•(b)は，現在メモリ上にあるプログラムを "test3.n88" というファイル名でドライブ2に入っているフロッピーディスクにアスキー形式でセーブします．

参照：LOAD, MERGE

SCREEN

スクリーン：screen

機　能

1） グラフィック画面に対しての種々のモードを設定する

2） グラフィックシンボルモードと日本語モードの切り替えを行う（N88-日本語 BASIC のみ）

書　式

1） SCREEN 〔〈画面モード〉〕〔,〈画面スイッチ〉〕〔,〈アクティブページ〉〕〔,〈ディスプレイページ〉〕……〈画面モード〉が0～2の場合

2） SCREEN 〔〈画面モード〉〕…… N88-日本語 で〈画面モード〉が3，4の場合

解　説

• N88-BASICでは，カラーモードにするか白黒モードにするか，白黒モードでどのページを書き込みおよび表示ページとするかなど，ディスプレイのグラフィック画面に対しての種々のモードを設定します．

• N88-日本語BASICでは，グラフィックシンボルモードと日本語モードとの切り替えを行います．

•〈画面モード〉は，カラー，白黒，分解能など，グラフィック画面の最も基本的なモードを設定するパラメータで次の値をとります．

0 —— カラーモード（640×200ドット）

1 —— 白黒モード（640×200ドット，3ページ）

2 —— 専用ディスプレイモード（640×400ドット）

• N88-日本語BASICでは，〈画面モード〉に3，4を指定することによって日本語モードにすることができます．

3 ——カラーモード（10行または12行表示，640×200ドット）

4 ——専用ディスプレイモード（20行または25行表示，640×400ドット）

なお，このとき〈画面スイッチ〉，〈アクティブページ〉，〈ディスプレイページ〉を指定することはできません．
日本語モードからグラフィックシンボルモードへの切り替えは，〈画面モード〉を0～2に指定することによって行います．

•〈画面スイッチ〉は，1ビットで表現される，"グラフィックマスクスイッチ"と"高速書き込みスイッチ"の2つのスイッチを合わせ持ったパラメータです．"グラフィックマスクスイッチ"をONにすると，現在グラフィック画面に描かれているパターンなどは一時的に消去されます．OFFにすれば元に戻ります．

• "高速書き込みスイッチ"は，CPUがグラフィックRAMに書き込むときのモードを設定

するもので，ONにすると通常より高速で書き込みます．

• N_{88}-BASIC V1（システムモードV1S）では，このスイッチをONにするとOFF時よりグラフィック動作が高速になります．ただし，N_{88}-BASIC V1（システムモードV1H）およびN_{88}-BASIC V2では，このスイッチは意味を持ちません．

• この2つのパラメータは次のように2ビットのビット対応になっており，与える値としては次のように0から3までとなります．

ビット1　ビット0

グラフィックマスクスイッチ→□□←高速書き込みスイッチ

0 ──→ [0][0]　高速書き込み，グラフィックマスクスイッチともOFF.

1 ──→ [0][1]　高速書き込みスイッチのみON.

2 ──→ [1][0]　グラフィックマスクスイッチのみON.

3 ──→ [1][1]　高速書き込み，グラフィックマスクスイッチともON.

• 〈アクティブページ〉と〈ディスプレイページ〉は白黒3ページモードの際の書き込むページと表示ページの選択であり，カラーモードと専用ディスプレイモードでは意味を持ちません．

• 〈アクティブページ〉は今後グラフィック命令によって書き込むページを0から2で指定します．〈ディスプレイページ〉は3ページのうちどのページを表示するかを指定します．この指定は各ページがビットに対応した3ビットの2進数表現（0〜7）で行います．

ビット2	ビット1	ビット0
ページ3	ページ2	ページ1

0 ──→ [0][0][0]　全ページ表示しない．

1 ──→ [0][0][1]　ページ1のみ表示．

2 ──→ [0][1][0]　ページ2のみ表示．

3 ──→ [0][1][1]　ページ1とページ2を合成して表示．

4 ──→ [1][0][0]　ページ3のみ表示．

5 ──→ [1][0][1]　ページ1とページ3を合成して表示．

6 ──→ [1][1][0]　ページ2とページ3を合成して表示．

7 ──→ [1][1][1]　全ページを合成して表示．

• この〈アクティブページ〉と〈ディスプレイページ〉の設定は気を付けないと，間違ってページを設定したり，書き込んだはずなのに表示ページになっていないなどのため，思ったとおりに表示できないことがあります．

• SCREEN文を実行すると，現在設定されているウィンドウ，ビューポート，LPは初期状態に戻されます(LPの初期状態はワールド座標，スクリーン座標とも(0，0)です)．

例　SCREEN 0,3

• 画面モードをカラーモードに設定し，画面スイッチをONに設定します．

プログラム例

• このプログラムは，N88-日本語BASICで実行してください．

```
100 ' SCREEN ｻﾝﾌﾟﾙ
110 SCREEN 0,0:GOSUB *DISPLAY
120 SCREEN 1,0,0,7:GOSUB *DISPLAY
130 SCREEN 2,0:GOSUB *DISPLAY
140 SCREEN 3:GOSUB *DISPLAY
150 SCREEN 4:GOSUB *DISPLAY
160 END
170 *DISPLAY
180  FOR I=0 TO 6
190    LINE(I*90,0)-STEP(89,199),I+1,BF
200  NEXT I
210  FOR J=0 TO 1000:NEXT J
220  CLS 2
230 RETURN
Ok
```

• カラーモード，白黒モード，専用ディスプレイモードで7本のカラーバーを描きます．

• 110行から150行で，それぞれの画面モードを設定しサブルーチンに実行を移します．

• 170行から230行で，7本のカラーバーを描きます．ただし，白黒モードおよび専用ディスプレイモードでは，カラーではなく白いバーが描かれます．

注意： • 白黒モード(専用ディスプレイモードを含む)において，グラフィック命令で〈パレット番号〉を指定した場合は，0か0以外かによって黒か白かを判定します．また〈パレット番号〉を省略した場合は，カラーモードと同様，現在COLOR文によって設定されている〈フォアグラウンドカラー〉のパレット番号が採用されます．その際も0か0以外かの判定により白，黒を決定します．

参照：プログラマーズガイド　第8章「画面モード」

SEARCH

サーチ：search

機能 任意の配列変数の要素の中から指定された値を探し出し，その要素ナンバを与える

書式 SEARCH(〈配列変数名〉,〈整数表記〉〔,〈開始ナンバ〉〕〔,〈ステップ値〉〕)

解説 •〈配列変数名〉で指定された配列変数の要素の中から〈整数表記〉で指定された値を探し，最初に見つかった要素ナンバを返します．

•指定された値が見つからなかった場合は，要素ナンバの代わりに−1を返します．

•〈配列変数名〉で指定する配列変数は，整数型の一次元配列でなければなりません．また配列変数は，SEARCH関数に先だってDIM文によって宣言しておかねばなりません．

•〈整数表記〉は，配列要素の中から探したい値を指定します．

•〈開始ナンバ〉には，配列内データのどこから探し始めるかを要素ナンバで指定します．省略した場合は，最初から探し始めます．

•〈ステップ値〉を指定した場合，そのステップをもって探し出します．省略した場合は1が指定されます．

•SEARCH文は，ランダムアクセスファイルのインデックスの検索や，GET@文，PUT@文で用いるデータの検索などに使うことができます．

例 NUM＝SEARCH(A%,100,0,3)

•A%という配列変数の中から100という数値を，要素ナンバ0から，要素ナンバ3つおきに探し出します．探し出した値は変数NUMに代入されます．

プログラム例

```
100 ' SEARCH ｻﾝﾌﾟﾙ
110 DIM A%(30)
120 I=1
130 A%(I)=I^3
140 I=I+1:IF I<=30 THEN 130
150 FIND%=7^3
160 X=SEARCH(A%,FIND%)
170 PRINT "FIND% =";FIND%,"X =";X
180 END
Ok
run⏎
FIND% = 343    X = 7
Ok
```

•配列変数A%の中から7^3を探し出すプログラムです．

•120行から140行で，配列変数A%にデータを代入します．

•160行で該当する数値を探し出し，170行で，その数値と要素ナンバを表示します．

SET

ディスク

セット：set

機能 ファイルまたはフロッピーディスクの属性を設定する

書式 SET 〈ドライブ番号〉,〈"属性文字"〉

SET #〈ファイル番号〉,〈"属性文字"〉

SET 〈ファイル名〉,〈"属性文字"〉

解説 • 指定したファイルやフロッピーディスクに，〈"属性文字"〉によって指定された属性を付けます．属性文字は大文字の "P", "R" で指定し，それぞれライトプロテクト（書き込み禁止），リードアフターライト（書き込み確認）の属性を付けます．これ以外の文字が属性文字として指定された場合には，現在設定されている属性がすべて解除されます．

• 〈ドライブ番号〉が指定された場合には，指定されたドライブの中に入っているフロッピーディスクに対して属性が付けられます．

• 〈ファイル番号〉が指定された場合には，以後そのファイルへの出力に対して指定された属性が作用します．

• 〈ファイル名〉が指定された場合には，指定されたファイルのみに属性が付けられ，同一のフロッピーディスク中の他のファイルにはその影響は及びません．

• P属性を付けると，PRINT#文，PUT文などの書き込み動作が禁止されるとともに，ファイルの削除もできなくなります．

• R属性を付けると，書き込みを行った直後に読み出しを行い，正しく書き込みが行われたかの確認がされるようになります．確認の結果，正しくなかった場合は "Disk I/O error" エラーになります．

例 SET 2,"P"

• ドライブ2に入っているフロッピーディスクに書き込み禁止の属性を付けます．

実行例

ディスク

```
set "2:test.n88","P"⏎
Ok
kill "2:test.n88"⏎
File write protected
Ok
```

• 実行例で，初めにドライブ2のフロッピーディスクに入っている "test.n88" というファイルにSET文でライトプロテクトをかけているため，KILLコマンドではファイルを消すことができません．

参照：ATTR$

SGN

エス・ジー・エヌ：sign

機能 数値の符号を返す

書式 SGN(〈数式〉)

解説
- 〈数式〉が正の場合は1を，0の場合は0を，負の場合は−1を返します．
- 〈数式〉には，整数型，単精度実数型，倍精度実数型が指定できます．

例

SGN(0.5) 返される値 ⇒ 1

SGN(0) 返される値 ⇒ 0

SGN(−0.5) 返される値 ⇒ −1

プログラム例

```
100 ' SGN ｻﾝﾌﾟﾙ
110 SCREEN 0,0:CLS 3
120 LINE(0,100)-(639,100),1
130 X=5
140 FOR I=0 TO 31.42 STEP .05
150   S=SGN(INT(COS(I)*20))
160   ON S+2 GOSUB 200,220,240
170   X=X+1
180 NEXT I
190 END
200 'プラスの場合
210 PSET(X,20),5:RETURN
220 'ゼロの場合
230 LINE(X,20)-(X,179),5:RETURN
240 'マイナスの場合
250 PSET(X,179),5:RETURN
Ok
```

- 矩形波を描きます．
- 140行からのFOR文の制御変数Iは，角度(ラジアン)を表します．
- 150行で余弦の値が正か負かを調べ，正なら画面の上方，負なら画面の下方に点を打ちます．

SIN

サイン：sine

機能 正弦(サイン)の値を返す

書式 SIN(〈数式〉)

解説
- 〈数式〉の値の正弦(サイン)を返します.
- 〈数式〉の値の単位はラジアン(π/180×角度)です.
- 〈数式〉に倍精度実数が含まれる場合は倍精度の値を返しますが，他の場合には単精度の値を与えます.

例

SIN(3.14159/180＊30) 返される値 0.5(SIN(30°)の値)

SIN(3.14159/180＊60) 返される値 0.866025(SIN(60°)の値)

プログラム例

```
100 ' SIN ｻﾝﾌﾟﾙ
110 F$="CSEC(###°)=###.#####"
120 INPUT "角度 :";D
130 R=D MOD 180
140 IF R=0 THEN PRINT "定義されません":END
150 C=1/SIN(R/180*3.1416)
160 PRINT USING F$;D,C
170 END
Ok
run↵
角度 :? 15↵
CSEC( 15°)=   3.86369
Ok
run↵
角度 :? 60↵
CSEC( 60°)=   1.15470
Ok
```

- 角度を度で読み込んで，コセカント(CSEC)の値を出力するプログラムです.
- 150行で

$$\mathrm{csec}\,\theta = 1/\sin\theta$$

を用いて計算しています.

参照：COS, TAN, ATN

SPACE$

スペース・ダラー：space $

機能 指定した長さの空白文字列を返す

書式 SPACE$(〈数式〉)

解説 •〈数式〉の数だけの空白を持った文字列を返します．〈数式〉の値の範囲は0～255です．

例 SPACE$(7) 返される値 "□□□□□□□"

• 7個の空白を持った文字列を返します．

プログラム例

• このプログラムは，N88-日本語BASICではグラフィックシンボルモード(SCREEN文の第1パラメータを0，1，2のいずれかにする)で実行してください．

```
100 ' SPACE$ ｻﾝﾌﾟﾙ
110 WIDTH 80,25
120 FOR I=-6 TO 6
130   A$=CHR$(&H88)+SPACE$(I^2)+"*"
140   PRINT A$
150 NEXT I
160 END
Ok
run
|                                    *
|                         *
|                *
|         *
|    *
| *
|*
| *
|    *
|         *
|                *
|                         *
|                                    *
Ok
```

• "＊" で放物線を描くプログラムです．

• 130行で

"｜"＋"Iの2乗個の空白"＋"＊"

を変数A$に代入しています．Iは−6から6まで変化するため，変数A$を順に出力すると，"＊" が放物線を描いているように見えます．

参照：SPC

SPC

エス・ピー・シー：space

機能 指定した数だけ空白を出力する

書式 SPC(〈数式〉)

解説
• SPC関数は，PRINT文などの出力文中でのみ使用することができます．

• 〈数式〉の値は−32768から32767まで許されますが，負の数はすべて0と見なされます．また，正の数でも現在の水平表示文字数以上の場合は，〈数式〉をその水平表示文字数で割った余りを値とします．

例 SPC(7) 返される値 "   "

• ただし，PRINT文などの出力文中で使用します．

プログラム例

• このプログラムは，N88-日本語BASICではグラフィックシンボルモード(SCREEN文の第1パラメータを0，1，2のいずれかにする)で実行してください．

```
100 ' SPC ｻﾝﾌﾟﾙ
110 WIDTH 80,25
120 FOR I=-6 TO 6
130   PRINT CHR$(&H88);SPC(I^2);"*"
140 NEXT I
150 END
Ok
```

• SPACE$関数のプログラム例と同じ動作をします．

• SPACE$関数の例では，連結した文字列をA$に代入してから出力していましたが，ここでは"|"と空白と"*"を順に直接画面に出力しています．

参照：SPACE$，TAB

SQR

スクウェア・ルート：square root

機能 平方根(スクウェアルート)を返す

書式 SQR(〈数式〉)

解説
- 〈数式〉で指定された値の平方根を返します.
- 〈数式〉に倍精度実数が含まれる場合は倍精度の値を返しますが，他の場合には単精度の値を返します.
- 〈数式〉の値が負のときは "Illegal function call" エラーとなります.

例

SQR(2) 返される値 1.41421

SQR(−2) 返される値 "Illegal function call" エラー

プログラム例

```
100 ' SQR ｻﾝﾌﾟﾙ
110 PRINT SPC(3)"<< 平方根  表 >>"
120 FOR N=0 TO 10
130   PRINT "N =";N,"ルート(N) =";
140   PRINT SQR(N)
150 NEXT N
160 END
Ok
run⏎
   << 平方根  表 >>
N = 0          ルート(N) = 0
N = 1          ルート(N) = 1
N = 2          ルート(N) = 1.41421
N = 3          ルート(N) = 1.73205
N = 4          ルート(N) = 2
N = 5          ルート(N) = 2.23607
N = 6          ルート(N) = 2.44949
N = 7          ルート(N) = 2.64575
N = 8          ルート(N) = 2.82843
N = 9          ルート(N) = 3
N = 10         ルート(N) = 3.16228
Ok
```

- 0から10までの平方根を出力します.

STOP

ストップ：stop

機　能　プログラムの実行を停止してコマンドレベルに戻る

書　式　STOP

解　説

• STOP文はプログラムの実行を停止するもので，プログラム中のどこに置いてもかまいません．

• STOP文を実行すると，次のメッセージが表示されコマンドレベルに戻ります．

Break in xxxxx

（xxxxxは停止した行の行番号）

• END文と異なり，STOP文はファイルを閉じません．

• CONTコマンドにより，停止したSTOP文の次からプログラムの実行を再開することができます．

例　STOP

• プログラムの実行を停止させます．

プログラム例

```
100 ' STOP ｻﾝﾌﾟﾙ
110 I=10
120 PRINT I,SQR(I)
130 STOP
140 I=I+1
150 GOTO 120
Ok
run⏎
 10            3.16228
Break in 130
Ok
```

• このプログラム例は130行がなければ，10以上の整数とその平方根を次々と出力するものです．

• 実行されると，10とその平方根を出力してすぐ停止します．STOP文による停止なので，CONTコマンドを入力することによって，STOP文の次の140行から実行を続けることができます．

参照：CONT

STOP ON/OFF/STOP

ストップ・オン/オフ/ストップ：stop on/off/stop

機能 ［停止］による割り込みの許可，禁止，停止を設定する

書式
1) STOP ON
2) STOP OFF
3) STOP STOP

解説
1) 割り込みを許可します．
- 以後，［停止］が押されるごとに，ON STOP GOSUB文によって設定された処理ルーチンに分岐します．

2) 割り込みを禁止します．
- 以後，［停止］を押しても，割り込み処理ルーチンへ分岐は起こりません．

3) 割り込みを停止します．
- 以後，［停止］を押しても，押されたことを覚えているだけで，処理ルーチンへ分岐は起こりません．しかし，STOP ON文によって割り込みが許可されると，先ほどの［停止］を押されたことによる割り込みで処理ルーチンへ分岐します．
- プログラムの終了時には，必ずSTOP OFF文を実行してください．

例 STOP ON
- ［停止］による割り込みを許可します．

プログラム例

```
100 ' STOP ON/OFF/STOP ｻﾝﾌﾟﾙ
110 ON STOP GOSUB *MASK
120 STOP ON
130 F$="###     ##.######^^^^  ##.######^^^^"
140 PRINT "角度";TAB(12);"Sin";TAB(27);"Cos"
150 FOR I=0 TO 180 STEP 20
160   TH=I/180*3.14159
170   PRINT USING F$;I;SIN(TH);COS(TH)
180 NEXT I:STOP OFF
190 END
200 *MASK:RETURN
Ok
run⏎
角度        Sin            Cos
  0      0.000000E+00   1.000000E+00
 20      3.420200E-01   9.396930E-01
 40      6.427870E-01   7.660450E-01
 60      8.660250E-01   5.000010E-01
 80      9.848080E-01   1.736490E-01
100      9.848080E-01  -1.736470E-01
120      8.660270E-01  -4.999980E-01
140      6.427890E-01  -7.660430E-01
160      3.420230E-01  -9.396920E-01
180      2.808800E-06  -1.000000E+00
Ok
```

• 0から20度ごとに，180度までの正弦(SIN)と余弦(COS)を計算します．

• このプログラム例は，ON STOP GOSUB文と同じです．

参照：ON STOP GOSUB, KEY ON/OFF/STOP

STR$

エス・ティー・アール・ダラー：string $

機　能　数値を文字列に変換した値を返す

書　式　STR$(〈数式〉)

解　説　•〈数式〉によって指定された値を，文字列に変換した値を返します．〈数式〉の値はすべての数値型を使うことができます．

•変換された文字列の最初のキャラクタは符号です(正の数の場合は空白，負の数の場合は"−"となります)．

例

STR$(123)　返される値⇒　"□123"

STR$(−1.5E+10)　返される値⇒　"−1.5E+10"

プログラム例

```
100 ' STR$ ｻﾝﾌﾟﾙ
110 INPUT A#:IF A#<0# OR 1D+16=<A# THEN 110
120 A$=STR$(A#):L=LEN(A$)-1
130 K=L MOD 3:IF K=0 THEN K=3
140 B$=LEFT$(A$,K+1):A$=MID$(A$,K+2)
150 IF A$="" THEN 190
160 B$=B$+","+LEFT$(A$,3)
170 A$=MID$(A$,4)
180 GOTO 150
190 PRINT B$
200 END
Ok
run⏎
? 123456789⏎
 123,456,789
Ok
run⏎
? 123⏎
 123
Ok
run⏎
? 12345678901234567⏎
? 1234567890123456⏎
 1,234,567,890,123,456
Ok
```

•正整数を読み込んで，3桁ごとに","を付けて出力します．

•倍精度実数は16桁までは指数形式を用いないで表示できます．したがって，読み込んだ数値が16桁を超えた場合は，110行でもう一度読み込むようにしています．

•120行で，読み込んだ数値を文字列に変換して変数A$に代入します．

•130行で変数A$の内容を3桁ごとに区切った場合，上位の桁から見て，初めに","を付けるまでに何桁表示すればよいかを計算しています．結果は変数Kに代入されます．

• 140行で初めの上位K桁を変数B$に代入し，変数A$からそれを消去します．これで変数A$の内容の桁数は3の倍数になります．

• 150行から180行では，変数A$から3桁ずつ取り出して，","とともに変数B$の内容につなげています．

参照：VAL

STRING$

ストリング・ダラー：string $

機　能　指定した文字からなる指定した長さの文字列を返す

書　式　STRING$(〈式〉, |〈文字列〉|)
　　　　　　　　　　　　　|〈数式〉|

解　説

• 与える文字は，〈文字列〉または〈数式〉によって指定します．〈文字列〉の場合，最初の1文字が有効です．〈数式〉の場合はキャラクタコード(0～255の範囲の値)で指定します．

• N88-日本語BASICでは，文字として全角文字を指定することもできます．このとき，指定できる文字およびそのシフトJISコードの範囲は，CHR$関数で指定できる範囲と同じです．

例

STRING$(5, "＊")　返される値 ⇒ "＊＊＊＊＊"

STRING$(2, 75)　返される値 ⇒ "KK"

STRING$(2, &H889F)　返される値 ⇒ "亜亜"

プログラム例

```
100 ' STRING$ ｻﾝﾌﾟﾙ
110 INPUT "文字列を入力してください ";A$
120 L=LEN(A$)
130 PRINT STRING$(L+4,"+")
140 PRINT "+ ";SPC(L+1);"+"
150 PRINT "+ ";A$;" +"
160 PRINT "+ ";SPC(L+1);"+"
170 PRINT STRING$(L+4,"+")
180 END
Ok
run
文字列を入力してください ? STRING$ ｻﾝﾌﾟﾙ ﾌﾟﾛｸﾞﾗﾑ
++++++++++++++++++++++++
+                      +
+ STRING$ ｻﾝﾌﾟﾙ ﾌﾟﾛｸﾞﾗﾑ +
+                      +
++++++++++++++++++++++++
Ok
```

• 読み込んだ文字列を "＋" で囲んで出力するプログラムです．

参照：CHR$，資料4 「キャラクタコード」

SWAP

スワップ：swap

機能 2つの変数の値を交換する

書式 SWAP〈変数名1〉,〈変数名2〉

解説 • どの型の変数(整数，単精度，倍精度，文字，単純変数と配列変数にかかわらず)でも，SWAP文によりその値を入れ替えることができます．ただし，その2つの変数の型は一致していなければなりません．

• 〈変数名2〉が単純変数(配列変数以外の変数)のとき，その変数には値が定義されていなければなりません．〈変数名1〉の変数は，定義されていなくてもかまいません．

例 SWAP A$,B$

• A$とB$に入っている値を入れ替えます．

プログラム例

```
100 ' SWAP ｻﾝﾌﾟﾙ
110 A$="太郎":B$="花子":C$="愛しています":D$="嫌っています"
120 GOSUB 160
130 SWAP A$,B$:SWAP C$,D$
140 GOSUB 160
150 END
160 PRINT USING "&  &は&  &を&            &";A$,B$,C$
170 RETURN
Ok
run↵
太郎は花子を愛しています
花子は太郎を嫌っています
Ok
```

• "太郎は花子を愛しています"と，"花子は太郎を嫌っています"という文を出力するプログラムです．

• 130行で，"太郎"と"花子"，"愛しています"と"嫌っています"をそれぞれ入れ替えています．

TAB

タブ：tab

機能 指定された桁位置まで空白を出力する

書式 TAB(〈数式〉)

解説
- TAB関数はPRINT文などの出力文中のみで使用されます.
- 〈数式〉の値は－32768から32767(0が左端)まで許されますが，負の数はすべて0と見なされます.
- 〈数式〉の値が正の数の場合，現在の水平表示文字数で割った余りを値とします.

例 TAB(20) 返される値 (20桁目までの空白)

- ただし，PRINT文などの出力文中で使用します.

プログラム例

```
100 ' TAB ｻﾝﾌﾟﾙ
110 FOR DEG=0 TO 180 STEP 20
120   RAD=DEG/180*3.14159
130   PRINT "角度 = ";USING"###°";DEG;
140   PRINT TAB(15) "SIN";USING"(###°) = ";DEG;:PRINT SIN(RAD);
150   PRINT TAB(40) "COS";USING"(###°) = ";DEG;:PRINT COS(RAD)
160 NEXT DEG
170 END
Ok
run
角度 =   0°    SIN(  0°) =  0          COS(  0°) =  1
角度 =  20°    SIN( 20°) =  .34202     COS( 20°) =  .939693
角度 =  40°    SIN( 40°) =  .642787    COS( 40°) =  .766045
角度 =  60°    SIN( 60°) =  .866025    COS( 60°) =  .500001
角度 =  80°    SIN( 80°) =  .984808    COS( 80°) =  .173649
角度 = 100°    SIN(100°) =  .984808    COS(100°) = -.173647
角度 = 120°    SIN(120°) =  .866027    COS(120°) = -.499998
角度 = 140°    SIN(140°) =  .642789    COS(140°) = -.766043
角度 = 160°    SIN(160°) =  .342023    COS(160°) = -.939692
角度 = 180°    SIN(180°) =  2.8088E-06 COS(180°) = -1
Ok
```

- 角度と，その正弦と余弦を出力するプログラムです.
- 140行と150行のTAB関数で，"SIN"と"COS"をそれぞれ15桁目と40桁目にそろえて出力させています.

参照：SPC

TAN

タンジェント：tangent

機能 正接(タンジェント)を返す

書式 TAN(〈数式〉)

解説
- 〈数式〉の値の正接(タンジェント)を返します.
- 〈数式〉の値の単位はラジアン(π/180×角度)です.
- 〈数式〉に倍精度実数が含まれる場合は倍精度の値を返しますが，他の場合には単精度の値を返します.

例

TAN(3.14159/180*30) 返される値 0.57735(TAN(30°)の値)

TAN(3.14159/180*60) 返される値 1.73205(TAN(60°)の値)

プログラム例

```
100 ' TAN ｻﾝﾌﾟﾙ
110 F$="COT(###°)=###.#####"
120 INPUT "角度 :";D
130 R=D MOD 180
140 IF R=0 THEN PRINT "定義されません":END
150 C=1/TAN(R/180*3.1416)
160 PRINT USING F$;D,C
170 END
Ok
run⏎
角度 :? 45⏎
COT( 45°)=   1.00000
Ok
run⏎
角度 :? 30⏎
COT( 30°)=   1.73205
Ok
```

- 角度を度で読み込んでコタンジェント(COT)を求めるプログラムです.
- 150行で

$$\cot\theta = 1/\tan\theta$$

を使って求めています.

注意：• TAN関数は

TAN(〈数式〉)＝SIN(〈数式〉)/COS(〈数式〉)

という式で算出しているため，COS関数の値が0となるとき(つまり，〈数式〉の値がπ/2, 3π/2, 5π/2, ...のとき)"Division by zero" エラーとなります.

参照：SIN, COS, ATN

T

TERM

ターミナル：terminal

機能 システムをターミナルモードにする

書式 TERM "〔COM:〕〔〈パリティ〉〔〈ビット〉〔〈ストップビット〉〔〈XONスイッチ〉〔〈Sパラメータ〉〕〕〕〕〕"〔,〔〈モード〉〕〔,〈変数領域の大きさ〉〕〕

解説

• 本機をBASICモードからターミナルモードに移します．

• ターミナルモードでは，RS-232Cインタフェースを介して他のコンピュータや機器との通信が可能となります．また，ターミナルモード時においても，リモートBASICプロトコルによりN_{88}-BASIC/N_{88}-日本語BASICの機能を利用することができます．詳しくは**プログラマーズガイド** 第17章「コンピュータコミュニケーション」を参照してください．

• 最初の〔COM:〕で，コミュニケーションポートを使って外部機器との入出力を行うことを宣言します．

• 〔COM:〕を省略した場合は，後に続くすべてのパラメータを指定しなければなりません．

• 〈パリティ〉はE，O，Nで表し，それぞれ偶数パリティ，奇数パリティ，パリティなしを意味します．

• 〈ビット〉は1文字を表すのに用いるビット数で，7または8で指定します．7を指定した場合には，必ず〈パリティ〉でOまたはEを指定しなければなりません．また8を指定した場合には，〈パリティ〉はNでなければなりません．

• 〈ストップビット〉は，ストップビット数を決定し，1，2，3で指定します．1は1ビット，2は1.5ビット，3は2ビットを意味します．

• 〈XONスイッチ〉は，XON/XOFFによる通信制御を行うことを指定し，スイッチとしてXが指定されるとXON/XOFF制御を行います．Nが指定された場合にはこの制御は行われません．

• 〈Sパラメータ〉は，〈ビット〉に7を指定したときに，キャラクタコード128以上の文字(カナ文字など)の入出力ができなくなるのを補助するパラメータです．指定はSまたはNで行い，Sを指定したとき入出力可能で，Nを指定したときは入出力不可能になります．

• 〈モード〉はHまたはFで指定し，それぞれ半二重，全二重通信のモードを意味します．ただし，モデムやカプラを接続する場合は，調歩同期全二重方式のモデムやカプラを使用し，〈モード〉は必ずFに指定してください．半二重方式のモデム，カプラは使用できません．

• 〈変数領域の大きさ〉は，リモートBASICプロトコルを利用する際に用いられる変数領域の大きさを決定します．省略された場合1024に設定されます．

• ターミナルモードからBASICモードに戻るときには，［シフト］＋［停止］を押します．

• ダブルクォーテーション(")で囲まれた部分(ダブルクォーテーションを含む)は，

OPEN文によりRS-232Cコミュニケーションファイルを開く場合にも同一の書式で用いられます.

例

term "com:"

• ターミナルモードに入ります.

実 行 例

```
term "com:E72X",H⏎
```

• 偶数パリティ，データ長7ビット，ストップビット1.5ビット，Xパラメータ有効，半二重モードのターミナルモードになります.

注意：• N88-日本語BASICの日本語モードでこのコマンドを実行すると，グラフィックシンボルモードに変わった後，ターミナルモードになります.

この場合ターミナルモードからBASICモードに戻っても，グラフィックシンボルモードのままとなります.

• 〈変数領域の大きさ〉を除くすべてのパラメータは，省略された場合，リセットしたとき(電源投入時も同様)に設定されていたセットアップモードの値が採用されます(セットアップモードについては，**ユーザーズガイド**を参照してください). また，途中のパラメータを省略して後のパラメータを指定する場合は，省略記号としてブランク(" ")を指定しなければなりません.

• ここで使用する各パラメータは，必ず大文字を使ってください.

参照：プログラマーズガイド 第17章2 「ターミナルモードでの通信」

TIME$

タイム・ダラー：time $

機能 内蔵のカレンダ時計の時刻を返す

書式 TIME$

解説
- TIME$変数を参照することにより，内蔵カレンダ時計の時刻が得られます．
- 時刻は8文字の文字列で返され，文字変数に代入したりPRINT文で内容を表示することができます．
- 時刻を設定するには，TIME$変数に次の形式の文字列を代入します．

"HH : MM : SS"

HH …時(00～23)
MM…分(00～59)
SS …秒(00～59)
各2文字で指定します．

例

TIME$ 返される値 (現在の時刻)

TIME$＝"21 : 11 : 00"

- 内蔵カレンダ時計に21時11分を設定します．

プログラム例

```
100 ' TIME$ ｻﾝﾌﾟﾙ
110 WIDTH 40,20:CLS
120 LOCATE 13,7
130 PRINT "只今の時刻"
140 LOCATE 14,9
150 PRINT TIME$
160 GOTO 140
Ok
```

- 画面上に時刻を表示させます．

参照：DATE$

TIME$ ON/OFF/STOP

タイム・ダラー・オン/オフ/ストップ：time $ on/off/stop

機能 リアルタイムタイマによる割り込みの許可，禁止，停止を設定する

書式
1) TIME$ ON
2) TIME$ OFF
3) TIME$ STOP

解説
1) 割り込み動作を許可します.
- この命令実行後は，設定された時刻になると割り込みが発生し，ON TIME$ GOSUB文によって定義された処理ルーチンに分岐します.

2) 割り込み動作を禁止します.
- この命令実行後は，割り込みが発生せず，処理ルーチンへの分岐は起こりません.

3) 割り込み動作を停止します.
- この命令実行後は，割り込みが発生したことを覚えているだけで，処理ルーチンへの分岐は起こりません.
- TIME$ ON文により割り込みが許可されると，先ほどの割り込みによって処理ルーチンへ分岐します.

- プログラム終了時には，TIME$ OFFを実行してください.

例 TIME$ ON
- リアルタイムタイマの割り込み動作を許可します.

プログラム例

```
100 ' TIME$ ON/OFF/STOP ｻﾝﾌﾟﾙ
110 ON TIME$="06:30:00" GOSUB *MORNING
120 CLS 3
130 TIME$ ON
140 LOCATE 30,10:PRINT TIME$
150 GOTO 140
160 *MORNING
170  TIME$ OFF
180  FOR I=1 TO 10
190    BEEP:PRINT "おはよう !!"
200    FOR J=0 TO 300:NEXT J
210  NEXT I
220 END
Ok
```

- 6時30分になると，ビープ音とともに"おはよう!!"を10回表示するプログラムです.
- このプログラム例は，ON TIME$ GOSUB文と同じです.

参照：ON TIME$ GOSUB, TIME$

TRON/TROFF

トレース・オン/トレース・オフ：trace on/trace off

機能　実行トレースモードを制御する

書式　TRON

TROFF

解説　・プログラムのデバッグのために実行トレースモードがあります．トレースモードでプログラムを実行すると，実行されている行の行番号が表示されます．

・TRONコマンドで実行トレースモードに入ります．TROFFコマンドは，それを解除するコマンドです．NEWコマンドを実行した場合も，実行トレースモードは解除されます．

例　TRON

・実行トレースモードに入ります．

実行例

```
100 ' TRON/TROFF ｻﾝﾌﾟﾙ
110 I=1
120 PRINT I
130 I=I+1
140 IF I<=3 THEN 120
150 END
Ok
tron⏎
Ok
run⏎
[100][110][120] 1
[130][140][120] 2
[130][140][120] 3
[130][140][150]
Ok
troff⏎
Ok
run⏎
 1
 2
 3
Ok
```

・TRONコマンドを使って，上記のプログラムの実行されている行番号を表示しています．TROFFコマンドにより，通常の実行状態に戻っています．

USR

ユーザ：user

機能 機械語で作られたユーザ関数ルーチンを呼び出し，その値を返す

書式 USR〔〈番号〉〕(〈引数〉)

解説
- USR関数は，ユーザの作成した機械語関数ルーチンを呼び出します．
- USR関数は，DEF USR文で実行開始番地を設定した上で使用します．
- 〈番号〉は0～9の値で，DEF USR文により定義された番号に対応します．〈番号〉が省略された場合は0と見なされます．
- 〈引数〉により，BASICから機械語ルーチンへの値の受け渡しをすることができます．

例 USR3(0) 返される値 ⇒ (USR3 関数を呼び出して返される値)

プログラム例

- このプログラムはN_{88}-BASICで実行してください．

```
100 ' USR ｻﾝﾌﾟﾙ
110 CLEAR ,&HDFFF
120 DEF USR=&HE000:GOSUB 210
130 CONSOLE 0,25,0,1:WIDTH 80,25:CLS
140 FOR I=1 TO 80*24
150   PRINT "♥";
160 NEXT I
170 CLS:FOR I=1 TO 500:NEXT I
180 VAD%=&HF3C8
190 I=USR(VAD%):FOR I=1 TO 500:NEXT I
200 END
210 FOR AD=&HE000 TO &HE015
220   READ DA:POKE AD,DA
230 NEXT AD
240 RETURN
250 '
260 DATA &H7E            :'       LD   A,(HL)
270 DATA &H23            :'       INC  HL
280 DATA &H66            :'       LD   H,(HL)
290 DATA &H6F            :'       LD   L,A
300 DATA &H3E,&HE9       :'       LD   A,"♥"
310 DATA &H11,&H28,&H00  :'       LD   DE,40
320 DATA &H0E,&H19       :'       LD   C,25
330 DATA &H06,&H50       :' L1:   LD   B,80
340 DATA &H77            :' L2:   LD   (HL),A
350 DATA &H23            :'       INC  HL
360 DATA &H10,&HFC       :'       DJNZ L2
370 DATA &H19            :'       ADD  HL,DE
380 DATA &H0D            :'       DEC  C
390 DATA &H20,&HF6       :'       JR   NZ,L1
400 DATA &HC9            :'       RET
Ok
```

U

• BASICと機械語のプログラムで，テキスト画面全体に"♥"を2回描きます.

• このプログラム例は，DEF USR文と同じものです.

参照：DEF USR, CALL

VAL

バリュー：value

機能 文字列の表す数値を返す

書式 VAL(〈文字列〉)

解説
- 〈文字列〉を数値に変換した値を返します．
- 〈文字列〉の最初の文字が"＋"，"－"，"&"，"."，"□"，または数字でなければVAL関数の値は0になります．また，数字以外の文字(ただし，16進数の場合の"A"～"F"，&Hの"H"，&Oの"O"，および指数表現の"D"と"E"を除く)が含まれている場合，それ以降の文字は無視されます．
- 文字列中のスペースは無視します．

例 VAL("123") 返される値 123

VAL("&HFF") 返される値 255(＝&HFF)

プログラム例

```
100 ' VAL ｻﾝﾌﾟﾙ
110 T$=TIME$
120 H=VAL(LEFT$(T$,2))
130 M=VAL(MID$(T$,4,2))
140 S=VAL(RIGHT$(T$,2))
150 SS=H*3600+M*60+S
160 PRINT "00:00:00 から ";T$;" までは";
170 PRINT SS;"秒です"
180 END
Ok
run
00:00:00 から 15:12:04 までは 54724 秒です
Ok
```

- 00：00：00から，現在のTIME$変数の時刻までの秒数を出力するプログラムです．
- 120行から140行で，時，分，秒を表す文字列を数値に変換しています．

参照：STR$

VARPTR

バリアブル・ポインタ：variable pointer

機能

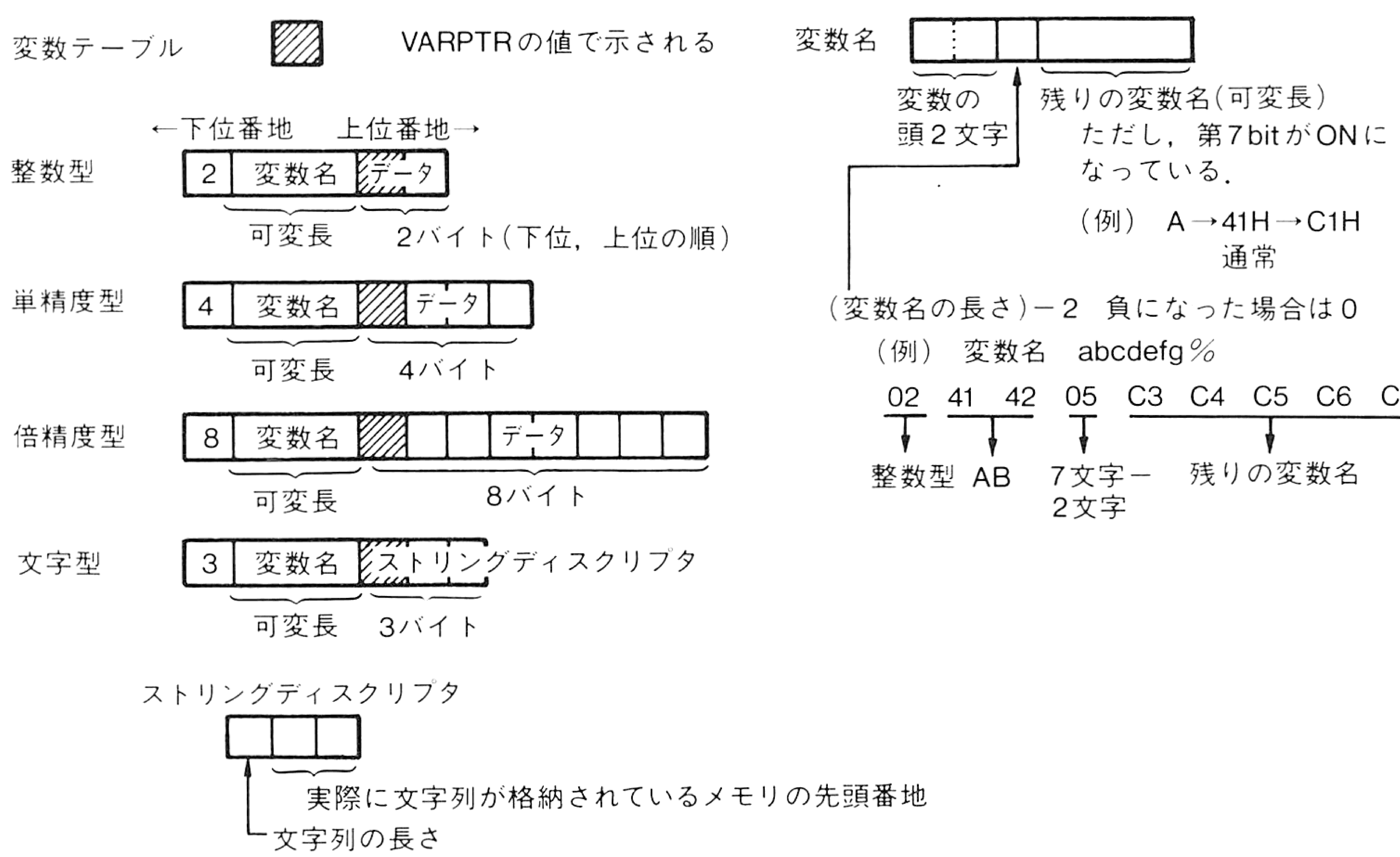
変数の格納されているメモリ番地，ファイルに割り当てられているバッファの先頭番地を返す

書式

VARPTR（〈変数名〉）
VARPTR（#〈ファイル番号〉）

解説

•引数として〈変数名〉を指定すると，その変数のデータが格納されている変数領域のメモリ番地を返します．このとき指定される変数は値が代入されていなければなりません．



•引数として〈ファイル番号〉を指定すると，その〈ファイル番号〉に割り当てられている入出力バッファの先頭番地−9を返します．

例

VARPTR(#1) 返される値 (入出力バッファ1の先頭番地−9)

プログラム例

```
100 ' VARPTR ｻﾝﾌﾟﾙ
110 ' value ¥ 2
120 DEFINT A
130 INPUT "整数値を入力してください ";A
140 ADR=VARPTR(A)
150 AL=(A¥2) MOD 256
160 AH=(A¥2)¥256
```

```
170 POKE ADR,AL
180 POKE ADR+1,AH
190 PRINT A
200 GOTO 130
Ok
run⏎
整数値を入力してください ? 1245⏎
 622
整数値を入力してください ? 510⏎
 255
整数値を入力してください ? 9035⏎
 4517
整数値を入力してください ? [停止]
Break in 130
Ok
```

• 整数値を読み込んで，その$\frac{1}{2}$の値を出力します．小数点以下は切り捨てます．

• 140行で，実数Aの値が格納されているアドレスを変数ADRに代入します．

• 150行と160行で上位バイトと下位バイトを別々に求めて，170行と180行で直接値を書き替えています．

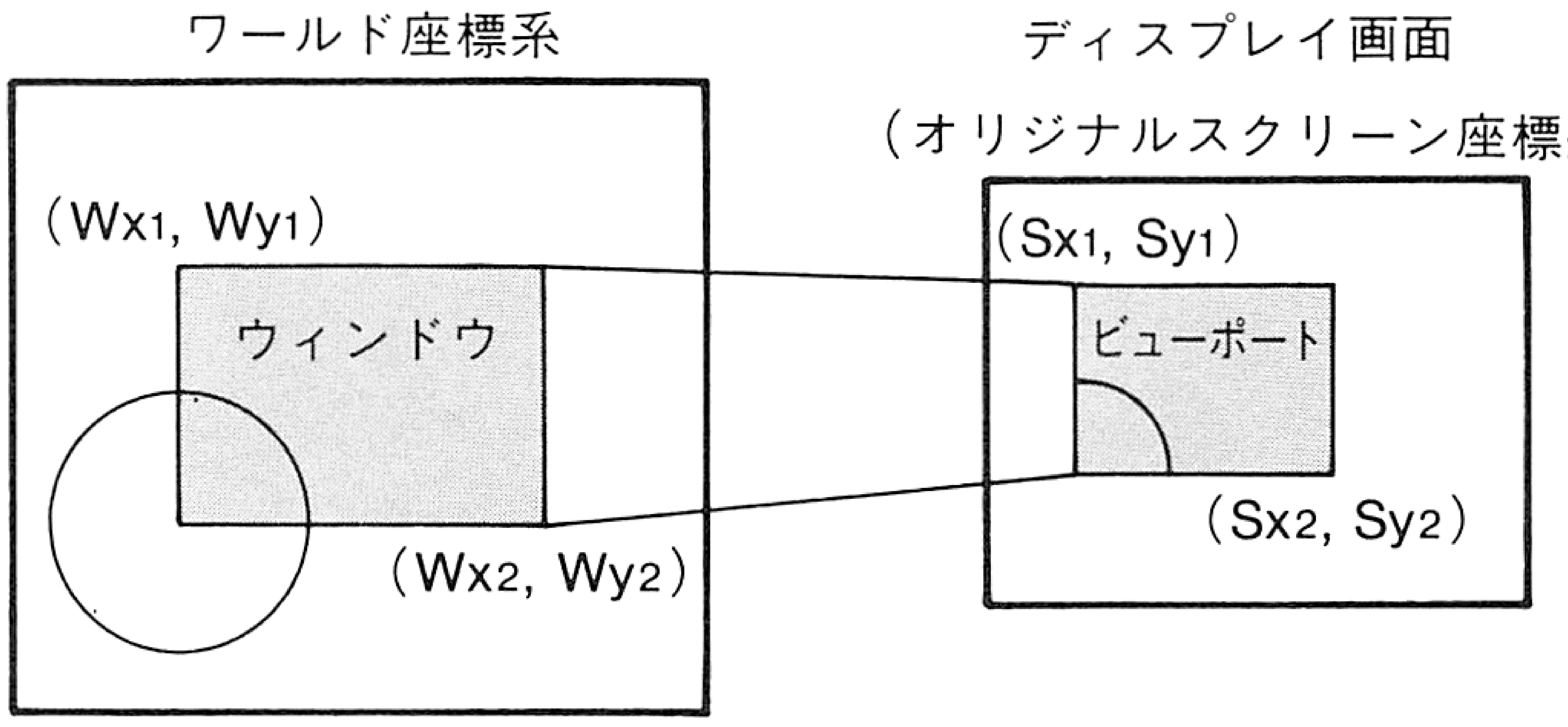
VIEW

ビュー：view

機能 ディスプレイ画面上での表示領域(ビューポート)を指定する

書式 VIEW (Sx_1, Sy_1) − (Sx_2, Sy_2) 〔,〈**領域色**〉〕〔,〈**境界色**〉〕

解説 •(Sx_1, Sy_1)を左上の頂点，(Sx_2, Sy_2)を右下の頂点とするオリジナルスクリーン座標系上の長方形を図形表示領域として指定します．オリジナルスクリーン座標とは，ディスプレイ画面のドットと1対1に対応する物理的な座標のことです．WINDOW文によって指定されているワールド座標系上の領域内に入る図形は，VIEW文で指定された領域中(ビューポート)に表示されるようになります．



•〈領域色〉が指定された場合には，指定されたパレット番号でビューポート内を塗りつぶします．

•〈境界色〉が指定された場合には，指定されたパレット番号でビューポートの枠を描きます．CLS文によって消去される画面の範囲は，この枠によって囲まれた中だけで，枠は消されることはありません．

•VIEW文はグラフィック領域の指定を行うだけで，実際の画面に対する操作は行いません．

•$Sx_1 < Sx_2$, $Sy_1 < Sy_2$が成り立たない場合，あるいはこれらの座標がディスプレイ画面から外れている場合には，"Illegal function call" エラーとなります．

•LPは，ビューポート左上の頂点(Sx_1, Sy_1)に設定されます．

プログラム例

```
100 ' VIEW ｻﾝﾌﾟﾙ
110 SCREEN 0,0
120 WINDOW(0,0)-(639,199)
130 GOSUB *DRAW
140 VIEW(1,1)-(319,198),0,7
150 GOSUB *DRAW
160 VIEW(500,1)-(630,99),0,7
170 GOSUB *DRAW
180 END
190 *DRAW
200  CLS 3:LINE(50,50)-(600,150),5,BF
210  LINE(50,50)-(600,150),0
220 RETURN
Ok
```

- ビューポートを変化させながら，斜線の入った四角形を描きます．
- 140行および160行のVIEW文でビューポートを変化させています．
- 190行から220行で，斜線の入った四角形を描いています．

注意：• ワールド座標系がビューポート内に展開されるのはWINDOW文の実行後であり，VIEW文のみ実行してウィンドウは初期状態のままであれば，ビューポート内の座標は(ワールド座標)＝(スクリーン座標)となります．

参照：WINDOW, CLS, SCREEN

VIEW

ビュー：view

機能 ビューポートの設定位置を返す

書式 VIEW(〈機能〉)

解説 • VIEW文により設定されているビューポートの位置(Sx_1, Sy_1)－(Sx_2, Sy_2)を返します．

• 〈機能〉は0～3の整数値で指定します．

0：ビューポート左上の頂点のX座標(Sx_1)

1：ビューポート左上の頂点のY座標(Sy_1)

2：ビューポート右下の頂点のX座標(Sx_2)

3：ビューポート右下の頂点のY座標(Sy_2)

• VIEW関数は，ディスプレイ画面上のビューポートの位置を返す関数ですから，返される値はオリジナルスクリーン座標です．

例 VIEW(0) 返される値 ⇒ (ビューポート左上の頂点のX座標値)

プログラム例

```
100 ' VIEW ｻﾝﾌﾟﾙ
110 VIEW(10,20)-(30,40)
120 IF VIEW(0)=10 AND VIEW(1)=20 AND VIEW(2)=30 AND VIEW(3)=40
    THEN PRINT "ビュー機能は正常に働いています。"
130 END
Ok
run ⏎
ビュー機能は正常に働いています。
Ok
```

• VIEW(10, 20)－(30, 40)でビューポートを設定した後，VIEW関数で読み込んだ座標と指定した座標が一致していたら"ビュー機能は正常に働いています。"を出力します．

参照：VIEW, WINDOW

WAIT

ウェイト：wait

機能 入力ポートをモニタし，その間プログラムの実行を中断する

書式 WAIT 〈I/Oアドレス〉，〈式1〉〔，〈式2〉〕

解説 • WAIT文は，指定した入力ポートのビットパターンが指定した状態になるまでプログラムの実行を中断します．

• 指定したポートから読み込んだデータと〈式2〉とのXORの結果と〈式1〉とのANDが取られ，その結果が真(0以外)になるまでポートのデータを読み続けます．XORとANDの結果をAとして式で表せば次のようになります．

A＝((読み込んだデータ)XOR〈式2〉)AND〈式1〉

この結果(この場合A)が0(偽)であれば，BASICはもう一度ポートのデータを読み込み同じ操作を繰り返します．もし結果が0でなければ(真ならば)，プログラムの実行は次の文に移ります．

• 〈式2〉が省略された場合は0とみなします．

例 WAIT 4，8，255

• I/Oポートの4をモニタし，ビット3が0になるのを待ちます．つまり，"s"キーが押されるのを待ちます．

プログラム例

```
100 ' WAIT ｻﾝﾌﾟﾙ
110 PRINT "'s'キーを押すと開始します。"
120 WAIT 4,8,255
130 FOR I=&H21 TO &H4F
140   PRINT CHR$(I);
150 NEXT
160 END
Ok
run
's'キーを押すと開始します。
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNO
Ok
s
```

• "s" キーを押すまで待ち続けます．

• 120行で，"s"キーが押されるまで待ち続けます．

• "s"キーが押されるとキャラクタコードの一部を画面に表示します．

注意： • WAIT文は，式の指定を誤ると無限ループに入ってしまう場合があります．その場合には本機を再起動しなければなりません．

WHILE〜WEND

ホワイル・ダブルエンド：while〜wend

機能 条件が満足されている間繰り返し実行する

書式 WHILE 〈論理式〉

　≀

WEND

解説 •〈論理式〉が真である間，WHILE文とWEND文の間にある文の実行を繰り返します．

•〈論理式〉を評価した結果が真であれば，WHILE文とWEND文との間にある文を順番に実行し，WHILE文に戻ります．この繰り返しは，〈論理式〉が偽になるまで行われます．

•〈論理式〉を評価した結果が偽である場合は，WEND文に続く文に処理が移ります．

•〈論理式〉を評価した結果が最初から偽である場合は，WHILE文とWEND文の間にある文は一度も実行されません．

•WHILE〜WEND文はFOR〜NEXT文と同様に，入れ子構造にすることができます．この場合には，それぞれのWEND文は，それ以前の最も近くにあるWHILE文と対応すると解釈されます．

•WEND文はWHILE文と対になり，ループの終わりを示す働きをしますので，省略することはできません．

例 WHILE J<5

　≀

WEND

•Jが5以上になるまで，WHILE文とWEND文の間の文を実行します．Jの値が5になると，WEND文に続く文に制御を移します．

プログラム例

```
100 ' WHILE ~ WEND ｻﾝﾌﾟﾙ
110 I=1
120 WHILE I<6
130   PRINT I;"-";
140   J=1
150   WHILE J<=I
160     PRINT J;:J=J+1
170   WEND
180   PRINT:I=I+1
190 WEND
200 END
Ok
run⏎
 1 - 1
 2 - 1  2
 3 - 1  2  3
```

```
  4 - 1  2  3  4
  5 - 1  2  3  4  5
Ok
```

• 1から5までの数値を1ずつ足した結果を表示します.

• このプログラムは入れ子構造になっており，120行のWHILE文は190行のWEND文と150行のWHILE文は，170行のWEND文とそれぞれ対になっています.

参照：FOR～NEXT

WIDTH

ウィドゥス：width

機能 デバイスやファイルに対して文字の領域を設定する

書式
1) WIDTH〈桁数〉〔,〈行数〉〕
2) WIDTH〈ファイルディスクリプタ〉,〈サイズ〉
3) WIDTH #〈ファイル番号〉,〈サイズ〉

解説
1) テキスト画面に表示する文字の〈桁数〉と〈行数〉を設定します.

• 1行当たり40文字，または80文字の〈桁数〉を設定できます．また〈行数〉は20行か25行かのどちらかです.

• N₈₈-日本語BASIC日本語モードのSCREEN 3のときは，10行または12行で設定します.

•〈行数〉を省略した場合は，現在の行数のままで変化しません.

• WIDTH文を実行すると画面がクリアされ，スクロールウィンドウが初期化されます．WIDTH文はCONSOLE文の前に実行してください.

2) •〈ファイルディスクリプタ〉で指定されたデバイスに対してバッファの〈サイズ〉を設定します．ここで許されているデバイスは，コミュニケーションポートとプリンタです.

•〈サイズ〉の値は1から255まで許されています．たとえば，この命令がプリンタに対して実行されたとすれば，プリンタの1行に印字される文字数は〈サイズ〉で指定したものとなります．このようにプリンタに対して実行した場合，WIDTH LPRINTと同等の機能を得ることができます．初期状態では255に設定されています.

3) •〈ファイル番号〉で割り当てられているバッファ(チャネル)に対して，その大きさを〈サイズ〉で指定します.

•〈サイズ〉は1から255までの値です．以後この〈ファイル番号〉を使っているデバイスとの入出力は，この〈サイズ〉単位で行われます．初期状態では255に設定されています.

例 WIDTH 80,25

• テキスト画面を80桁，25行に設定します.

プログラム例

• このプログラムは，N₈₈-日本語BASICではグラフィックシンボルモード(SCREEN文の第1パラメータを0，1，2のいずれかにする)で実行してください.

```
100 ' WIDTH ｻﾝﾌﾟﾙ
110 WIDTH 80,20:CLS
```

```
120 GOSUB 170
130 IF INKEY$="" THEN 130
140 WIDTH 80,25:CLS
150 GOSUB 170
160 END
170 PRINT "┌────────┬────────┬────────┬────────┐"
180 PRINT "├────────┼────────┼────────┼────────┤"
190 PRINT "└────────┴────────┴────────┴────────┘"
200 RETURN
Ok
```

- 初めにテキスト画面を80桁×20行に設定して罫線を描き，次に80桁×25行で描きます．
- 110行から120行で，80桁×20行で罫線を描きます．この場合，線が接続されない部分があります．
- 130行は，80桁×20行で罫線を描いた後，次の罫線を描くのを待つためにあります．任意のキーを押すことによって次の処理に移ります．
- 140行から150行で，80桁×25行で罫線を描きます．このときは線がすべて接続されます．

参照：CONSOLE

WIDTH LPRINT

ウィドゥス・エルプリント：width line print out

機　能　プリンタに出力される１行の文字数を設定する

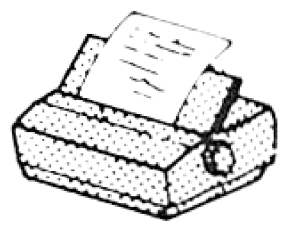
書　式　WIDTH LPRINT 〈文字数〉

解　説　
- プリンタの１行に印字することのできる文字数を設定します．
- 〈文字数〉で指定できる値は１から255までです．

例　WIDTH LPRINT 80

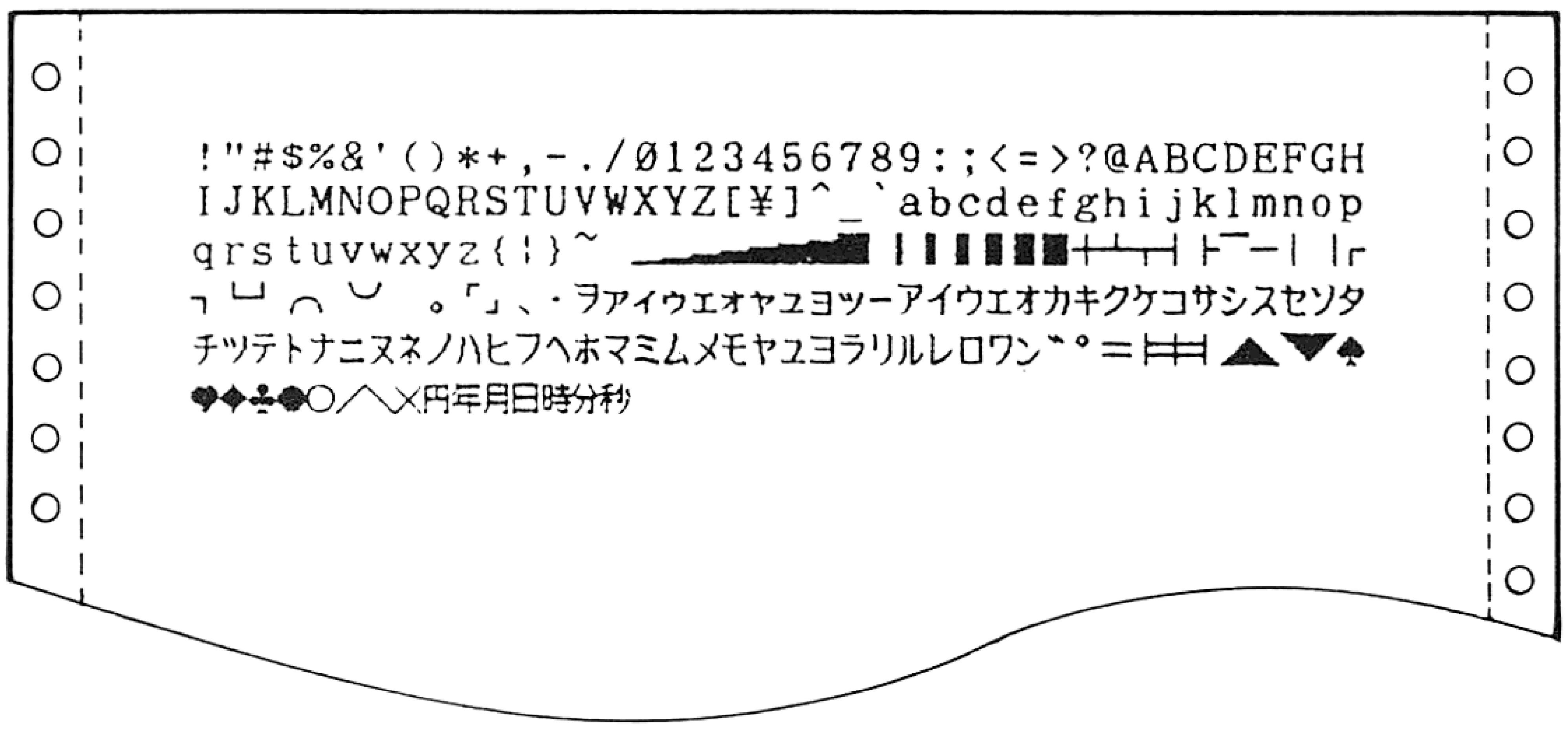
- プリンタの１行に印字する文字を80文字に設定します．

プログラム例



- このプログラムは，N₈₈-日本語BASICではグラフィックシンボルモード(SCREEN文の第１パラメータを0，1，2のいずれかに設定する)で実行してください．

```
100 ' WIDTH LPRINT ｻﾝﾌﾟﾙ
110 WIDTH "lpt:",255
120 WIDTH LPRINT 40
130 FOR I=33 TO 247
140   LPRINT CHR$(I);
150 NEXT I
160 LPRINT
170 END
Ok
run↵
Ok
```



- キャラクタコードの33から247までをプリンタに出力するプログラムです．
- 110行で，プリンタに対しバッファサイズを255に設定しています．
- 120行で，プリンタの１行に印字する文字数を40文字に設定しています．

参照：WIDTH 2)

WINDOW

ウィンドウ：window

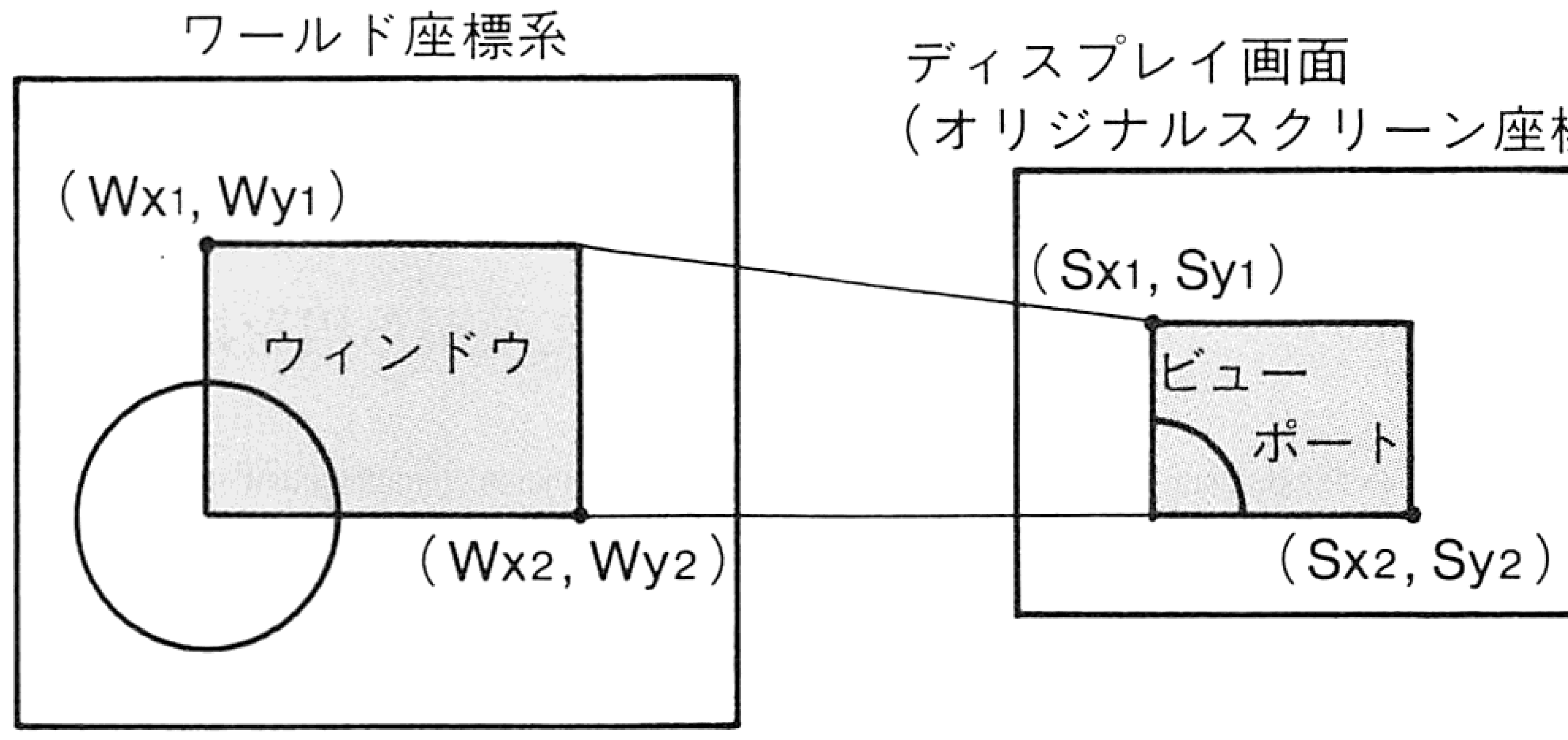
機能 ビューポートに表示するワールド座標系内の領域を指定する

書式 WINDOW $(Wx_1, Wy_1)-(Wx_2, Wy_2)$

解説 • (Wx_1, Wy_1)を左上の頂点，(Wx_2, Wy_2)を右下の頂点とするワールド座標系上の長方形で囲まれた領域を，スクリーン上のビューポートに表示する領域として指定します．

• WINDOW文実行後は，その領域内に入る図形はVIEW文で指定された領域（ビューポート）内に表示されます．

• WINDOW文で指定された領域はウィンドウと呼ばれ，そのウィンドウから外れた図形は表示されなくなります．



• WINDOW文もVIEW文と同じように領域の指定を行うだけで，実際の画面に対する操作は行いませんので，WINDOW文によりウィンドウの設定を変更するだけで，ビューポート中に表示される図形が変化することはありません．

• $Wx_1 < Wx_2$, $Wy_1 < Wy_2$が成り立たない場合，あるいはこれらの座標がワールド座標系から外れている場合には"Illegal function call"エラーとなります．

• WINDOW文で指定する座標はワールド座標ですから，負の値や実数値もとれます．

• 一度設定されたウィンドウは，次にWINDOW文あるいはSCREEN文が実行されるまで変化しません．

• WINDOW文が実行されると，LP(Last reference Point)はウィンドウの左上の頂点に設定されます．

• ワールド座標がビューポート内に展開されるのはWINDOW文の実行後であり，VIEW文のみ実行してウィンドウが初期状態のままであれば，ビューポート内の座標は，(ワールド座標)＝(スクリーン座標)となります．

例 WINDOW (−300, −50)−(100, 70)

• ウィンドウの左上の頂点を(−300, −50)に，右下の頂点を(100, 70)に設定します．

プログラム例

```
100 ' WINDOW ｻﾝﾌﾟﾙ
110 SCREEN Ø,Ø:CLS 3:C=1
120 VIEW(2ØØ,5Ø)-(4ØØ,15Ø),,7
130 WINDOW(-1ØØ,-1ØØ)-(1ØØ,1ØØ)
140 GOSUB *DRAW
150 WINDOW(-1ØØ,-1ØØ)-(Ø,Ø)
160 GOSUB *DRAW
170 WINDOW(Ø,Ø)-(1ØØ,1ØØ)
180 GOSUB *DRAW
190 WINDOW(-5ØØ,-5ØØ)-(5ØØ,5ØØ)
200 GOSUB *DRAW
210 END
220 *DRAW
230  CIRCLE(Ø,Ø),1ØØ,C
240  FOR J=Ø TO 1ØØØ:NEXT J
250  C=C+1:CLS 2
260 RETURN
Ok
```

- ウィンドウを4種類に設定しながら円を描くプログラムです.
- 140行，160行，180行，200行でGOSUB文を使用して，220行に実行を移しています.
- 220行以降のサブルーチンでは，色を変えながら円を描いています.

参照：SCREEN, VIEW, WINDOW

WINDOW

ウィンドウ：window

機能 現在のウィンドウの設定位置を返す

書式 WINDOW(〈機能〉)

解説

• WINDOW文によって設定されている現在のウィンドウの位置(Wx_1, Wy_1)－(Wx_2, Wy_2)を返す関数です.

• 〈機能〉は0から3までの値をとり，それぞれの値をとるときの関数値の意味は次のとおりです.

0: ウィンドウの左上の頂点のX座標(Wx_1)

1: ウィンドウの左上の頂点のY座標(Wy_1)

2: ウィンドウの右下の頂点のX座標(Wx_2)

3: ウィンドウの右下の頂点のY座標(Wy_2)

• WINDOW関数の返す値はワールド座標です.

例 WINDOW(1) 返される値⇒ ウィンドウの左上の頂点のY座標

プログラム例

```
100 ' WINDOW ｻﾝﾌﾟﾙ
110 SCREEN Ø,Ø
120 GOSUB *WINDOWINFO
130 VIEW(1ØØ,5Ø)-(3ØØ,15Ø)
140 GOSUB *WINDOWINFO
150 WINDOW(-1ØØ,-1ØØ)-(1ØØ,1ØØ)
160 GOSUB *WINDOWINFO
170 END
180 *WINDOWINFO
190  PRINT "WX1 :";WINDOW(Ø),
200  PRINT "WY1 :";WINDOW(1)
210  PRINT "WX2 :";WINDOW(2),
220  PRINT "WY2 :";WINDOW(3)
230  PRINT
240 RETURN
Ok
run⏎
WX1 : Ø        WY1 : Ø
WX2 : 639      WY2 : 199

WX1 : 1ØØ      WY1 : 5Ø
WX2 : 3ØØ      WY2 : 15Ø

WX1 :-1ØØ      WY1 :-1ØØ
WX2 : 1ØØ      WY2 : 1ØØ

Ok
```

• SCREEN文，VIEW文，WINDOW文をそれぞれ実行した場合のウィンドウの設定位置を返すプログラムです.

・120行，140行，160行で180行以降のサブルーチンに実行を移し，現在のウィンドウの設定位置を出力します．

参照：VIEW, WINDOW, MAP

WRITE

ライト：write

機能 画面へデータを出力する

書式 WRITE〔〈式の列〉〕

解説

• 〈式の列〉により指定された数値式，あるいは文字式の値を画面に出力します.

• 〈式の列〉が省略された場合には空行のみが出力されます.

• 式はコンマ(,)またはセミコロン(;)により区切ります．コンマとセミコロンとは機能上の区別はありませんが，画面上には各式の値の区切りとして必ずコンマが表示されます.

• WRITE文は，ほぼPRINT文と同じように画面表示を行いますが，不要な空白桁は詰められ，それぞれの式の値の間は必ずコンマで区切られます.

• 文字列はダブルクォーテーション(")で囲まれて出力されます.

例 WRITE I＊J，K，A$

• I＊J, K, A$の値をコンマで区切って画面に出力します.

プログラム例

```
100 ' WRITE ｻﾝﾌﾟﾙ
110 A%=12
120 B!=9.87654
130 C#=3.14159265359#
140 D$="N88-BASIC"
150 PRINT A%;B!,C#;D$
160 WRITE A%;B!,C#;D$
170 END
Ok
run⏎
 12  9.87654   3.14159265359 N88-BASIC
12,9.87654,3.14159265359,"N88-BASIC"
Ok
```

• まずPRINT文で変数の値を表示し，次にWRITE文で表示します.

• 160行のWRITE文では各値の間をコンマで区切り，文字列をダブルクォーテーションで囲んで出力します.

参照：PRINT, WRITE#

WRITE#

ディスク

ライト・シャープ：write #

機能 シーケンシャルファイルへデータを出力する

書式 WRITE #〈ファイル番号〉〔,〈式の列〉〕

解説
- 〈式の列〉により指定された数値式あるいは文字式の値を，〈ファイル番号〉により指定されたファイルに書き込みます．
- 指定されたファイルは出力用にオープンされていなければなりません．
- 〈式の列〉中の各式は，コンマ(,)あるいはセミコロン(;)によって区切ることができます．
- WRITE#文はほぼPRINT#文と同じように式の出力をしますが，不要な空白桁は詰められ，それぞれの式の値の間は必ずコンマで区切られます．
- 文字列はダブルクォーテーション(")で囲まれて出力されます．
- WRITE#文は各式の値を書き出した後，改行のコードを書き出します．〈式の列〉が省略された場合は，改行のコードのみを書き込みます．

例 WRITE #1,I*J,K;A$

- ファイル番号1のシーケンシャルファイルにI*J, K, A$の値を出力します．

プログラム例

ディスク

```
100 ' WRITE# ｻﾝﾌﾟﾙ
110 OPEN "data10" FOR OUTPUT AS #1
120  FOR I=1 TO 10
130    A$="レコード番号"
140    WRITE A$,I
150    WRITE #1,A$,I
160  NEXT I
170 CLOSE
180 END
Ok
run
"レコード番号",1
"レコード番号",2
"レコード番号",3
"レコード番号",4
"レコード番号",5
"レコード番号",6
"レコード番号",7
"レコード番号",8
"レコード番号",9
"レコード番号",10
Ok
```

• ドライブ１のフロッピーディスクに"data10"というシーケンシャルファイルを作ります．

• 120行から160行で"record number"を10個と，１から10までの数値を，画面と"data10"ファイルに出力します．

参照：PRINT#, WRITE

第3章

拡張命令

3.1 拡張命令の概要

N_{88}-BASIC V2/N_{88}-日本語BASICでは，拡張命令を追加することによって，次の2つの機能を使用することができます．

1. アナログパレット機能

　本体にアナログRGBディスプレイを接続することにより，512色の中から8色を選び表示することができます．この機能により色に濃淡をつけたり，中間色を用いることが可能です．

2. サウンド機能

　本体に内蔵されているシンセサイザーIC(OPN)を操作することにより，実際の楽器の音あるいは効果音などの中から選んで，同時に6音まで演奏することができます．既存の音または全く白紙の状態から，自分でアレンジして自分自身の音を作ることもできます．

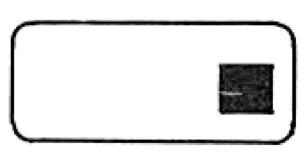
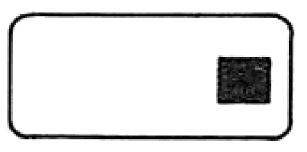
　内蔵のシンセサイザーIC(OPN)はFM音源3声とSSG音源3声の6声ですが，別売のサウンドボード(PC-8801-11)を接続することにより，FM音源6声にすることも可能です．これにより，より品質の高い音(FM音源6声)で演奏できます．

　また，別売のミュージックインタフェースボード(PC-8801-10)も動作可能ですので，キーボード等を接続して音楽を演奏することができます．

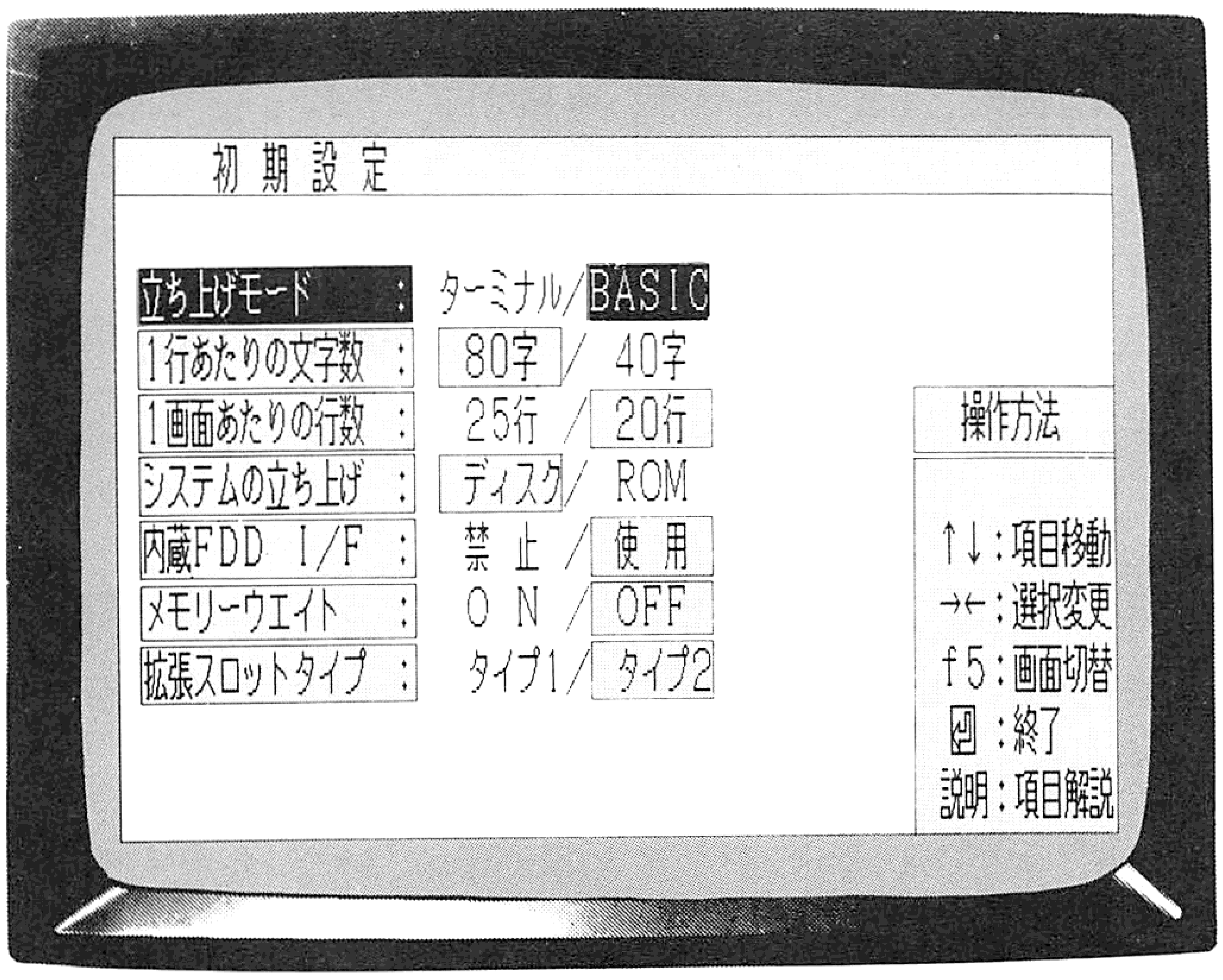
注意：本機にミュージックインタフェースボードを接続して使用する場合，ミュージックインタフェースボードのジャンパスイッチのジャンパコネクタは取り外してご使用ください．

3.2 拡張命令の追加

拡張命令はN_{88}-BASIC V2/N_{88}-日本語BASICで追加することができます．追加は，セットアップモードおよびスライドスイッチを設定したあと，NEW CMD文を実行することによって行います．(下表参照)

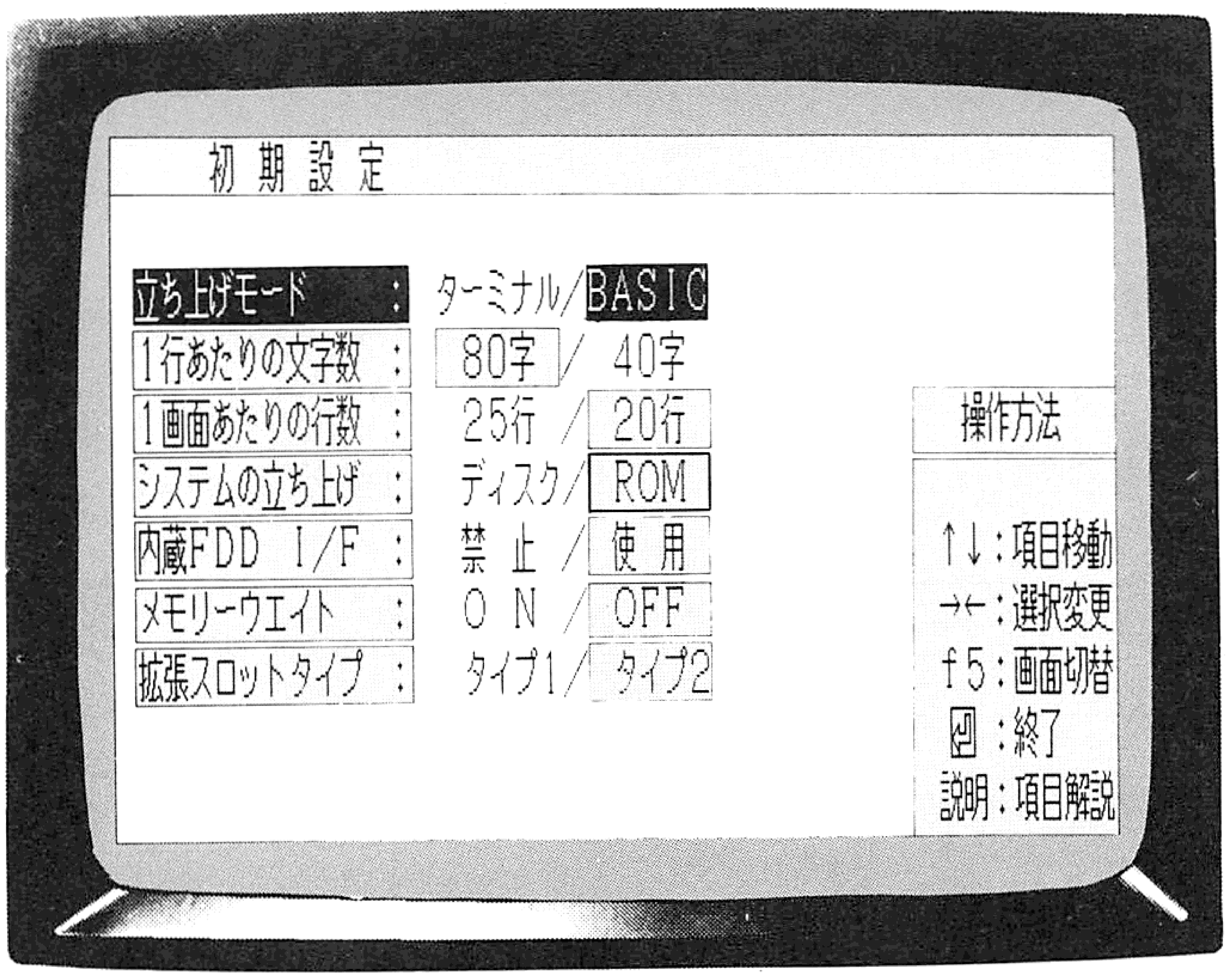
モデル名	セットアップモード画面の設定	BASICのモード	スライドスイッチの設定	ステートメント
PC-8801MH PC-8801FH モデル20 PC-8801FH モデル30	初期設定画面でフロッピーディスクからの起動ができるようにする--次ページ(a)	N_{88}-BASIC V2 N_{88}-日本語BASIC	 VIS V1H V2 システムモード	new cmd
PC-8801FH モデル10	初期設定画面でROMからの起動ができるようにする ――――――次ページ(b)	N_{88}-BASIC V2	 VIS V1H V2 システムモード	new cmd

(a)　セットアップモード画面〈PC-8801MH，PC-8801FHモデル20/30の場合〉



※　PC-8801MHの場合，拡張スロットタイプの項目はありません．

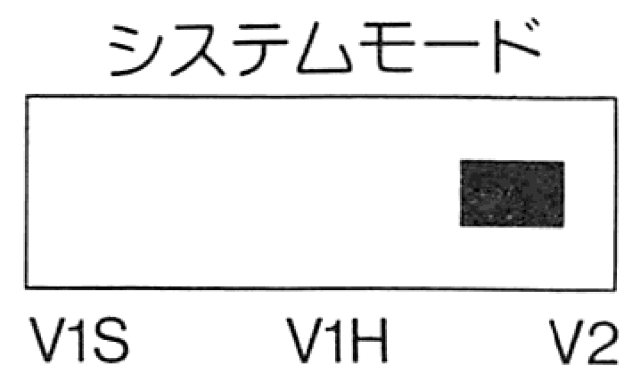
(b)　セットアップモード画面〈PC-8801FHモデル10の場合〉



拡張命令を追加する手順は次のとおりです．

注意： システムモードスイッチや動作クロックスイッチなど，スライドスイッチの切り替えは，必ず電源をOFFにしてから行ってください．

(1) システムモードスイッチをV2に合わせます.



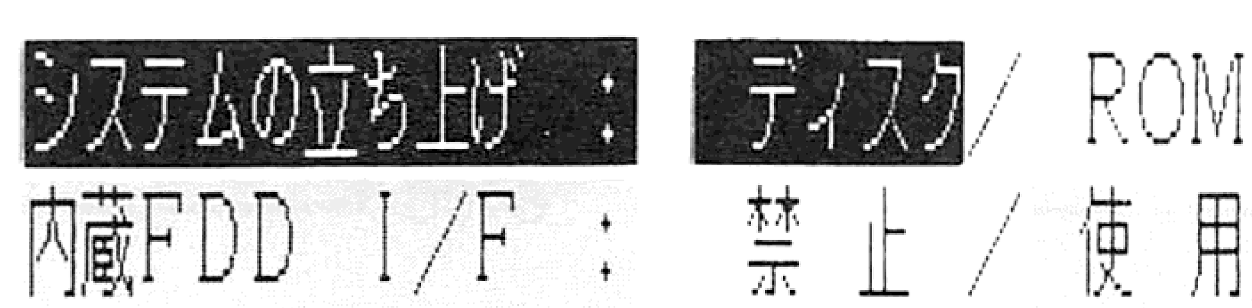
(2) 周辺機器から順番に電源を入れたあと，N_{88}-BASICシステムディスク，またはN_{88}-日本語BASICシステムディスクをドライブ番号1のフロッピーディスクドライブにセットします.

(3) PCキーを押しながらリセットボタンを押し，セットアップモードの起動を行います.

• V2 DISKの場合

初期設定画面で，フロッピーディスクが使用できるようにモードを設定します.

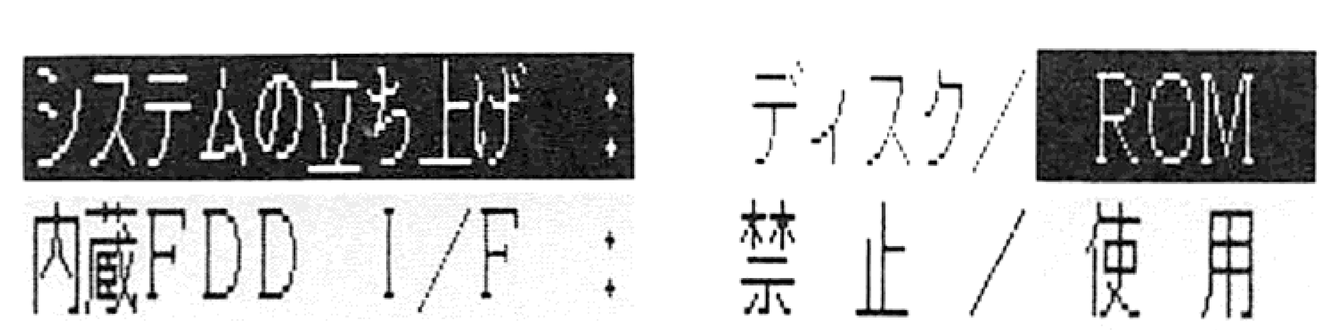
セットアップモードが終了すると，V2 DISKが起動します.(セットアップモードでの設定変更および終了方法は，**プログラマーズガイド** 第2章 「N_{88}-BASIC V2の起動/終了とモード設定」を参照)



• V2 ROMの場合

初期設定画面でフロッピーディスクの使用を中止するようにモードを設定します.

セットアップモードが終了すると，V2 ROMが起動します.(セットアップモードでの設定変更および終了方法は，**プログラマーズガイド** 第2章 「N_{88}-BASIC V2の起動/終了とモード設定」を参照)



(4) NEW CMD文を実行します.

new cmd ⏎

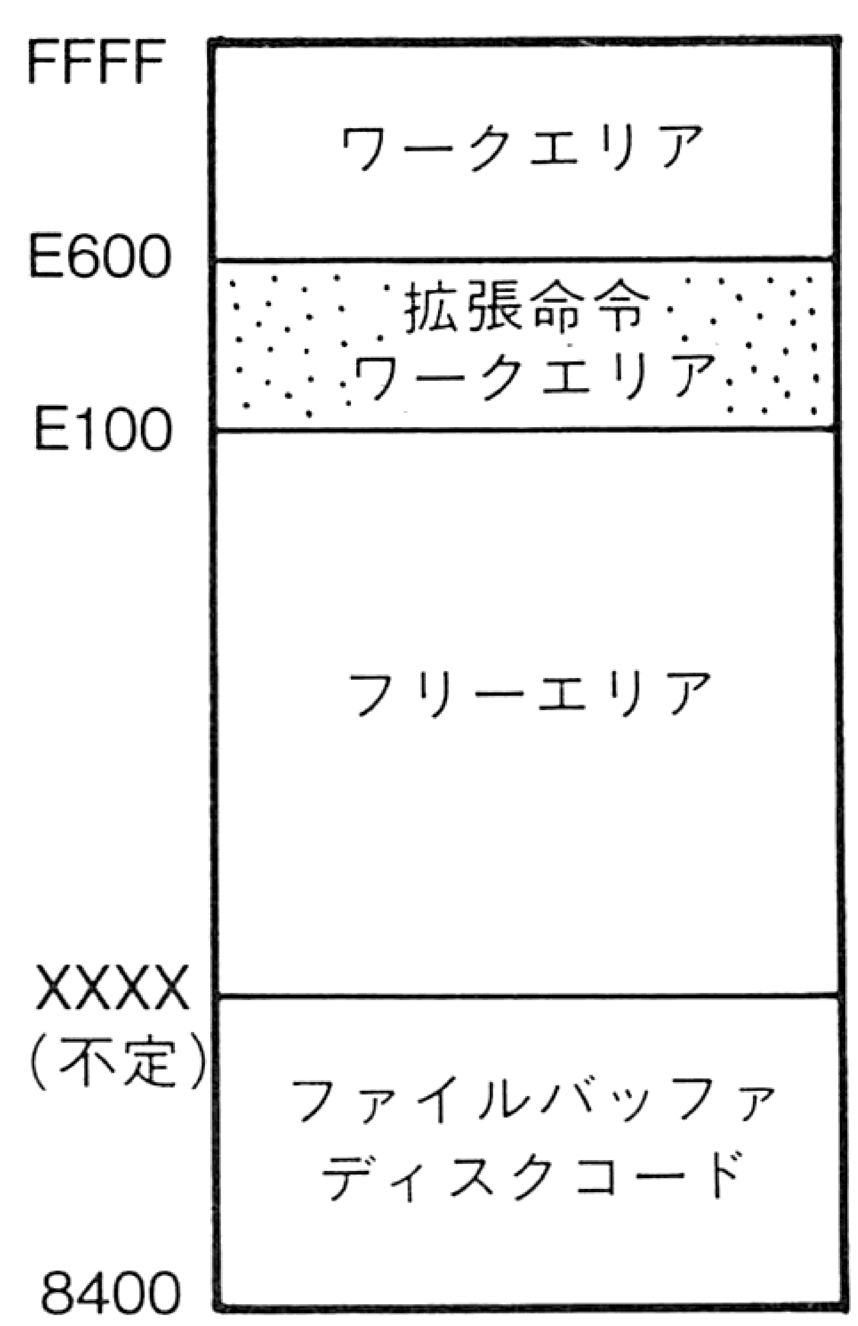
(5) 「Ok」と表示されると拡張命令が使用できます.

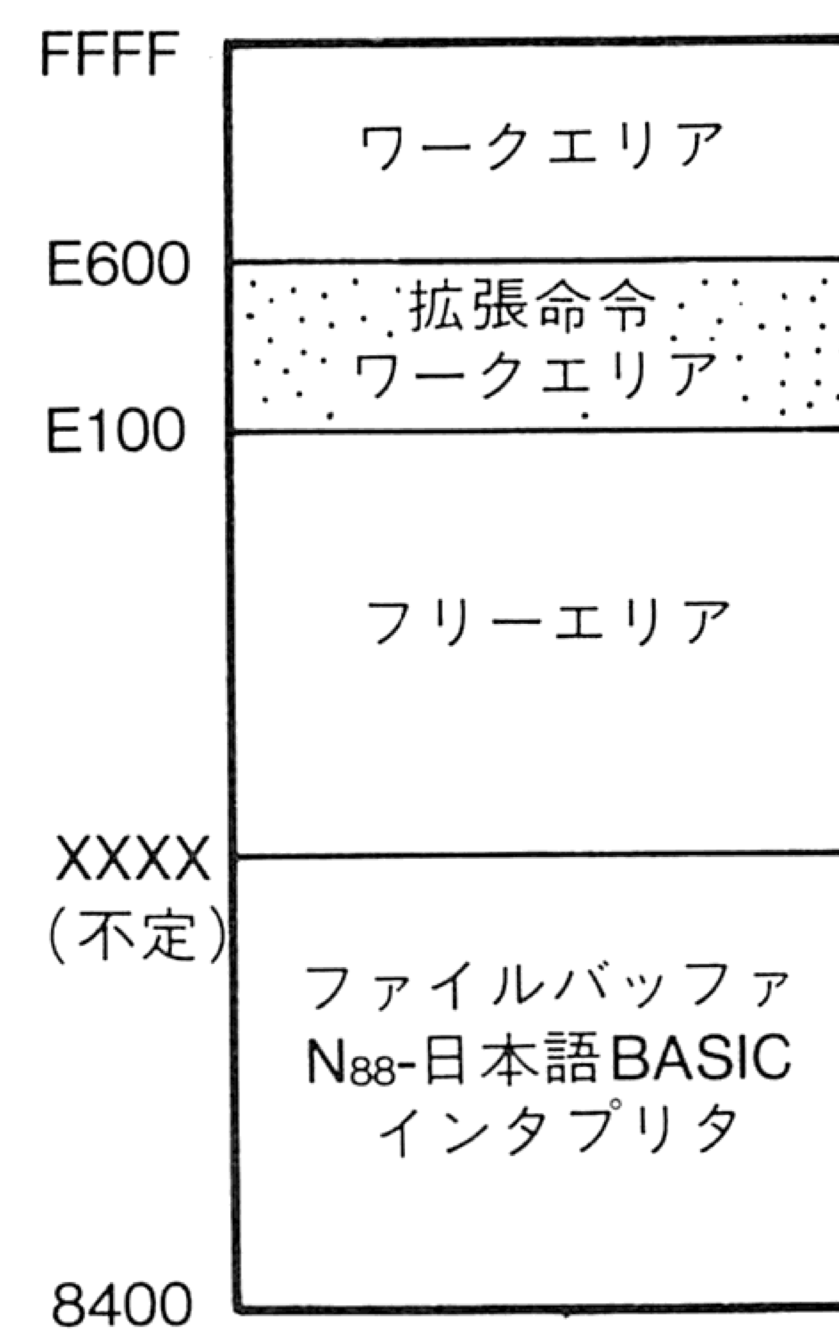
拡張命令はN_{88}-BASIC V1では使用できません. したがって，N_{88}-BASIC V1でNEW CMD文を実行すると"Syntax error"が表示されます. この場合，拡張命令は追加されていませんのでご注意ください.

注意： 拡張命令を追加した場合，ウォームスタート(停止 を押しながらリセットボタンを押す)を行うと，拡張命令はクリアされ使えなくなります. 再度追加する場合は，CMD UNLINK文を実行したあとにNEW CMD文を実行してください.

3.3　メモリマップ

NEW CMD文を実行すると，拡張命令のためのワークエリアがRAM領域にとられます．拡張命令を追加した場合のメモリマップは次のとおりです．





このように，E100H番地からE5FFH番地までワークエリアとして使用されています．したがって，機械語でプログラムを作る方は，この領域を避けた空間でプログラムが実行されるよう設計してください．

N_{88}-日本語BASICの場合も同様ですが，フリーエリアがN_{88}-BASICのときよりも少なくなりますので注意してください．

3.4 拡張ステートメント

CMD BGM

シー・エム・ディー・ビー・ジー・エム：cmd bgm

機能 並列動作モードの制御を行う

書式 CMD BGM 〈スイッチ〉

解説
- 並列動作を行うか行わないかを決めます.
- 〈スイッチ〉の値が1で並列動作モードになり，0で並列動作モードが解除されます.
- 並列動作モードは，CMD PLAY文などで音を出しながら次の命令を実行しますので，音を中断させることなく実行することができます.
- このモードを解除した状態では，音を出している間，次の命令の実行を待ちます.
- NEW CMD文を実行した直後のモードは，並列動作モード(ON)になっています.

例 CMD BGM 1
- 音楽機能を並列動作モードにします.

プログラム例

```
100 ' CMD BGM ｻﾝﾌﾟﾙ
110 CMD STOPM
120 CMD BGM 0
130 FOR I=0 TO 61
140   PRINT "音色番号 =";I
150   CMD PLAY "@=I;CDE"
160 NEXT
170 END
Ok
```

- 現在演奏中の音色番号を表示しながら音を出します.
- 120行で並列動作モードを解除しているため，150行のCMD PLAY文が終わるまでは次の音色番号が表示されません.
- 130行から160行までのFOR～NEXT文で，音色番号0から61の音色を使って"ドミソ"を演奏します.

注意：本機にミュージックインタフェースボード(PC-8801-10)を接続して使用する場合，ミュージックインタフェースボードのジャンパスイッチのジャンパコネクタは取り外してください.

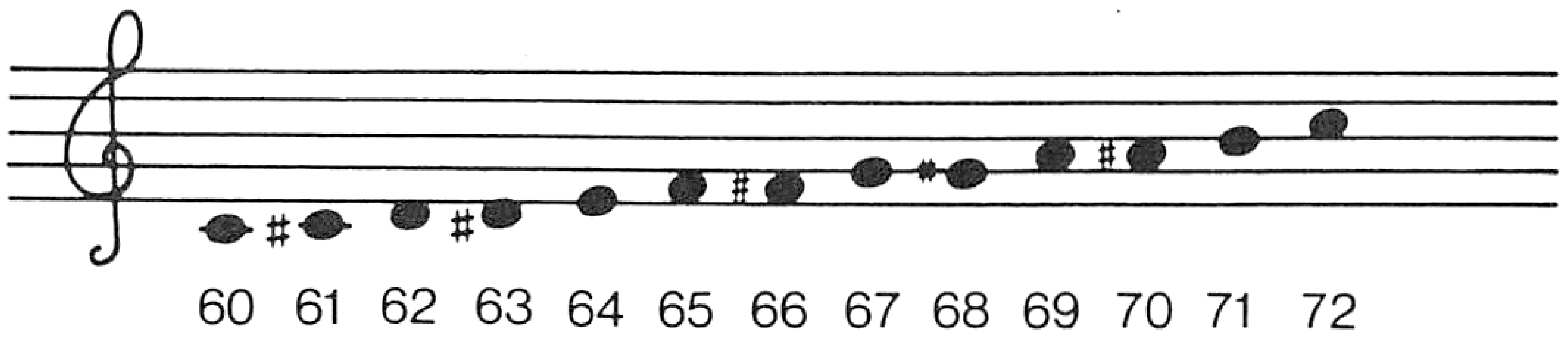
CMD OUTM

シー・エム・ディー・アウト・エム：cmd output midi

機能　MIDIポートへ１バイトのデータを送る

書式　CMD OUTM〈数式〉〔,〈数式〉,〈数式〉...〕

解説
- このステートメントは，本体にミュージックインタフェースボード(PC-8801-10)が接続してあるときに使用します．
- 音程は0～127の範囲で指定します．



半音上がるごとに＋1されます．第4オクターブのC(ド)の音は60の値になります．
- 音量は0～127の範囲で設定します．CMD PLAY文の"V"の値とCMD OUTM文の音量の対応は次のようになります．

CMD PLAY文の"V"の値	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CMD OUTM文の音量	8	16	24	32	40	48	56	64	72	80	88	96	104	112	120

- 詳しくは，ミュージックインタフェースボードに添付されているユーザーズマニュアルを参照してください．

例　CMD OUTM 60

- MIDIポートへ１バイトのデータ(60)を送り出します．

プログラム例

- このプログラムを実行する場合には，別売のミュージックインタフェースボードが必要です．

```
100 ' CMD OUTM ｻﾝﾌﾟﾙ
110 CMD OUTM &H90
120 CMD OUTM 60
130 CMD OUTM 64
140 FOR W=0 TO 500:NEXT W
150 CMD OUTM &H80
160 CMD OUTM 60
170 CMD OUTM 0
180 END
Ok
```

CMD PAL

シー・エム・ディー・パル：cmd palette

機能 パレットにアナログカラーコードを割り付ける

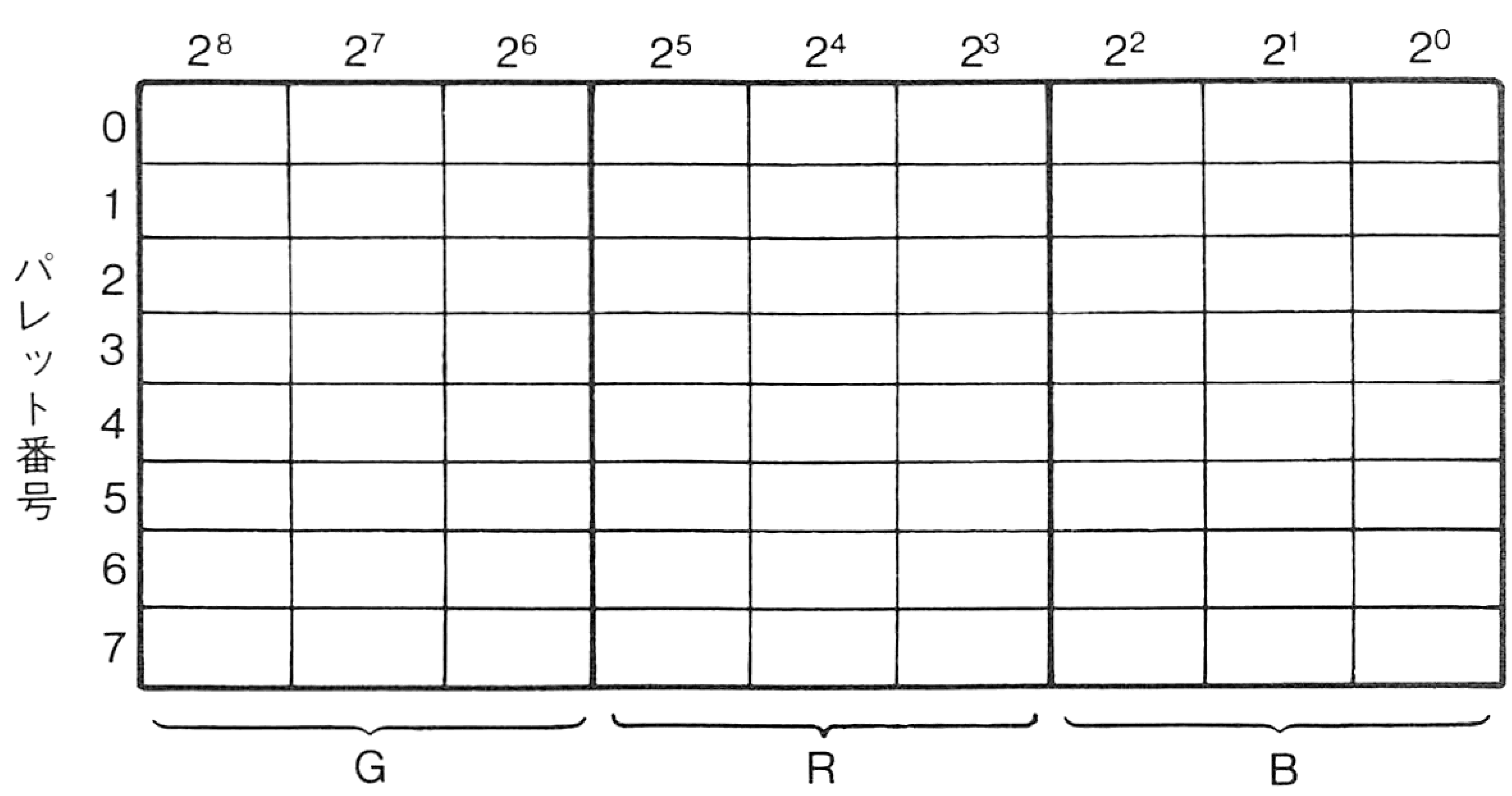
書式 CMD PAL〔〈パレット番号〉〕〔,〈アナログカラーコード〉〕

解説

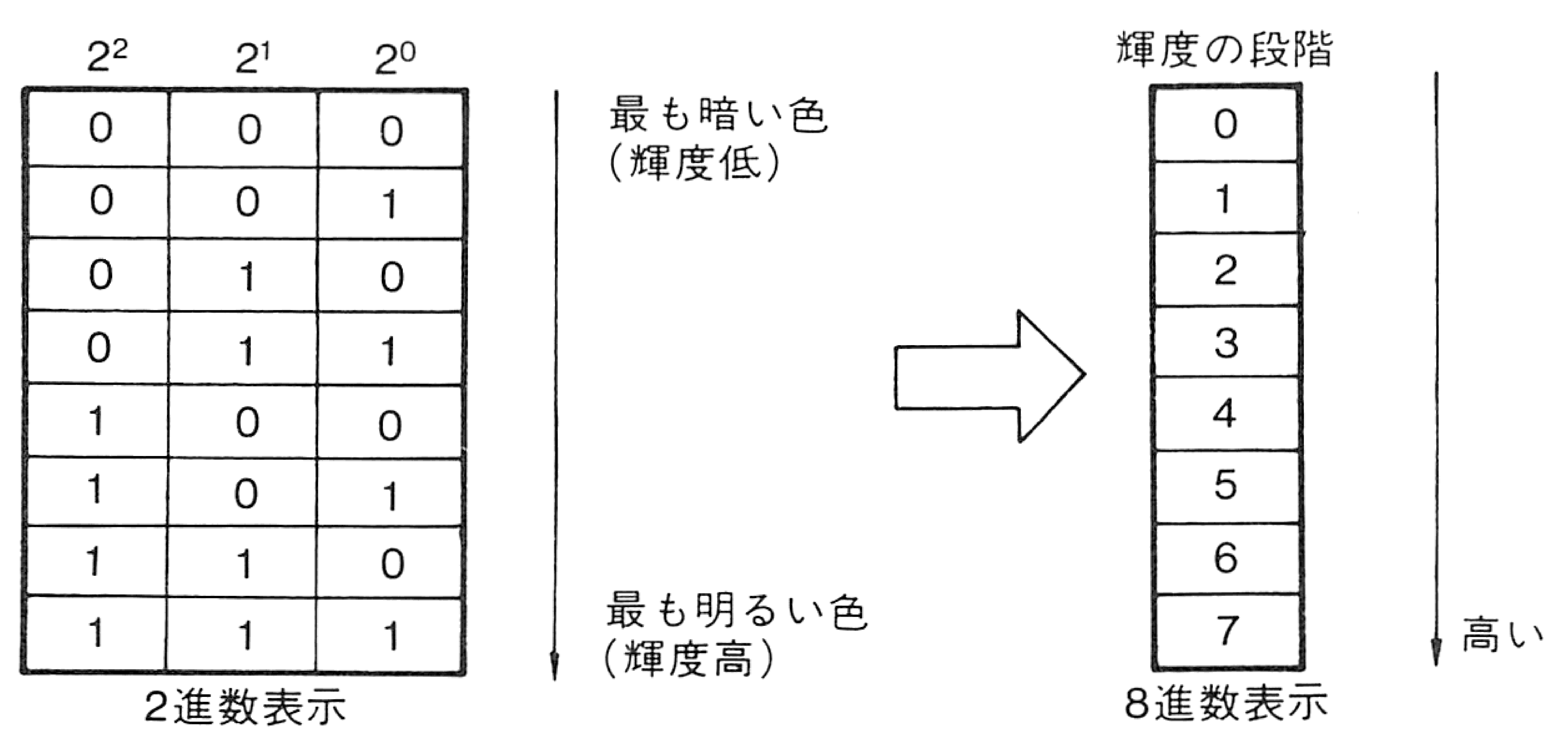
•〈パレット番号〉で指定されるパレットに，〈アナログカラーコード〉で指定される色を設定します.

•〈パレット番号〉は0〜7，〈アナログカラーコード〉は&O000〜&O777(10進で0〜511)の範囲の整数で設定可能です.

•パレットは全部で8個あり，1つのパレットは9ビットから成っています．9ビットは3ビットずつに区分され，下位3ビットは青(B)，中位の3ビットは赤(R)，上位の3ビットは緑(G)に対応しています．各3ビットは8段階の明るさを表しています.



例) B(青)のアナログカラーコード



このように1色につき8段階の輝度を表現することが可能なので，8進数を用いると便利で

す．たとえば，パレット番号１のパレットの初期状態(本体の電源を入れた直後またはリセットボタンを押した直後の状態)を図示してみると，次のようになっています．

	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
パレット番号 1	0	0	0	0	0	0	1	1	1
	G			R			B		

これは，GとRの色の輝度がゼロで，Bの色だけ輝度が111(２進数表現)つまり７(８進数表現)で"明るい青"を表すことになるわけです．上のパレットの状態をCMD PAL文で書くと次のようになります．

```
CMD PAL 1 , &O007
```

これと同様にパレット番号７のパレットの初期状態は，

```
CMD PAL 7 , &O777
```

と表せます．つまりR, G, Bすべての色の輝度がいちばん高くなっていて，"明るい白"を表現しているわけです．このようにR, G, Bは各８段階の輝度を表すことができますから，各パレットで表現できる色の数は8×8×8=512色となります．

• 初期状態のパレット番号とそれに対応する色の関係は次のようになります．

パレット番号	色	G			R			B			8進数表現		
		2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0			
0	黒	0	0	0	0	0	0	0	0	0	0	0	0
1	青	0	0	0	0	0	0	1	1	1	0	0	7
2	赤	0	0	0	1	1	1	0	0	0	0	7	0
3	紫	0	0	0	1	1	1	1	1	1	0	7	7
4	緑	1	1	1	0	0	0	0	0	0	7	0	0
5	水色	1	1	1	0	0	0	1	1	1	7	0	7
6	黄色	1	1	1	1	1	1	0	0	0	7	7	0
7	白	1	1	1	1	1	1	1	1	1	7	7	7

• CMD PAL文をパラメータなしで実行すると，各パレットの割り付けは初期状態に戻ります．

• CMD UNLINK文を実行すると拡張命令がキャンセルされ，CMD PAL文を使用することができなくなります．ただし，拡張命令がキャンセルされる前に設定したパレットの色は，拡張命令がキャンセルされた後でも有効となります．

例 CMD PAL 2，&O007

• パレット番号2に明るい青を設定します.

プログラム例

```
100 ' CMD PAL ｻﾝﾌﾟﾙ
110 SCREEN 0,0:CLS 3:CMD PAL
120 FOR X=0 TO 560 STEP 80
130   LINE(X,0)-(X+79,199),X/80,BF
140 NEXT X
150 FOR CR=0 TO 511
160   CMD PAL (CR MOD 8),CR
170   FOR I=1 TO 100:NEXT I
180 NEXT CR
190 CMD PAL
200 END
Ok
```

• 縦に8本のカラーバーを表示して，512色を出力するプログラムです.

• 160行で，それぞれのカラーバーに色を指定します.

• 190行で，パレットを初期状態に戻します.

注意：• CMD PAL文を実行する場合，本体にアナログRGBディスプレイを接続してください. デジタルRGBディスプレイが接続されていると，表示される色は黒，青，赤，紫，緑，水色，黄色，白の8色となります.

• アナログRGBディスプレイを接続してグラフィック命令(circle, line等)を実行した場合，多少縦長になる場合があります.

CMD PLAY

シー・エム・ディー・プレイ：cmd play

機　能　音楽を発生する

書　式　CMD PLAY〔#〈音源のモード〉,〕〔〈文字列1〉〕〔,〈文字列2〉〕〔,〈文字列3〉〕〔,〈文字列4〉〕〔,〈文字列5〉〕〔,〈文字列6〉〕

解　説

• 6和音まで演奏することができます．

•〈文字列 1〉，〈文字列 2〉，〈文字列 3〉はチャネル1，2，3に，〈文字列 4〉，〈文字列 5〉，〈文字列 6〉はチャネル 4，5，6 に対応します．

• ここでいうチャネルとは音源のことを意味します．

•〈音源のモード〉は 0～4 の値で指定します．この〈音源のモード〉をプログラミング時に変数にしておくと，切り替えが容易になります．

#0：別売のミュージックインタフェースボードのSSG音源を指定します．このボードにはPSGが 2 個実装されており，チャネル 1，2，3でPSG1 の各SSG音源を，チャネル 4，5，6 でPSG2 の各SSG音源を制御します．

PSG1			PSG2		
SSG音源			SSG音源		
チャネル1	チャネル2	チャネル3	チャネル4	チャネル5	チャネル6

#1：別売のミュージックインタフェースボードのMIDIインタフェースを指定します．この場合，MIDI仕様のシンセサイザが必要です．接続されたキーボードによりチャネルの最大数は異なりますが，CMD PLAY文では最大が 6 となります．

#2：省略時の場合の初期値です．内蔵シンセサイザIC(OPN)のFM音源，SSG音源を指定します．チャネル1，2，3が各FM音源を，チャネル 4，5，6 が各SSG音源を制御します．

OPN					
FM音源			SSG音源		
チャネル1	チャネル2	チャネル3	チャネル4	チャネル5	チャネル6

#3：OPNを効果音モードに切り替えます．CMD PLAY文中のYコマンドによってOPNのレジスタの内容を書き換えることにより音を出します．

#4： OPNをCSM(サイン波)モードに切り替えます．CMD PLAY文中のYコマンドによってOPNのレジスタの内容を書き換えることにより音を出します．

注意： 通常，〈音源のモード〉は#0，#1，あるいは#2でお使いください．#3，#4で使用する場合には，OPNの構造を十分に理解することが必要です(**プログラマーズガイド** 第12章5「FM音源の原理/シンセサイザICの構造」参照)．

• 各チャネルの文字列は，MML(Music Macro Language)といい，次のような意味を持ちます．

文　字	意　味	初期値
Mx（SSG音源のみ）	エンベロープ周期の設定.（1≦x≦65535）	M255
Sx（SSG音源のみ）	エンベロープ形状の設定.（0≦x≦15）	S1
Vx	音量の設定.（0≦x≦15）	V8
Lx	長さの設定.（1≦x≦64）	L4
Qx	音の長さの割合.（1≦x≦8）	Q8
Ox	オクターブの設定.（1≦x≦8）	O4
>	オクターブを1つ上げる.	
<	オクターブを1つ下げる.	
Nx	xで指定された高さの音を発生する.（0≦x≦96）	
Tx	テンポの設定.（32≦x≦255）	T120
Rx	休符の設定.（1≦x≦64）	R4
＋（#）	音を半音上げる.	
－	音を半音下げる.	
.	音符の長さや休符の長さを1.5倍にする.	
&	タイ．前後の音を継なぐ.	
{}x	連符．指定された長さのX分音符を{ }の中の音程の個数で等分にした音を発生する.	
@x（FM音源のみ）	xで指定された音色番号に切り替える.（音色番号表参照）	
Yr, d（OPNのみ）	OPNのレジスタrの内容をdにする.	
Zd（MIDIのみ）	MIDIにデータdを送る.	
@Vx（FM音源, MIDIのみ）	音量を細かく調整する.（0≦x≦127）	
@Wx	xで指定された長さだけ状態を維持する.（1≦x≦64）	@W4

• 初期値を持っているパラメータはCMD PLAY文実行中 [停止] で止めたり，"Illegal function call"エラー等でストップすると，初期値に戻ります．

1. 音程(Cx, Dx, Ex, Fx, Gx, Ax, Bx)

音程はC～Bで表し，パラメータxは音の長さを表します．その対応は次のとおりです．パラメータxが省略されたときは，Lコマンドの値が設定されたものとみなします．

文字	C	D	E	F	G	A	B
音程	ド	レ	ミ	ファ	ソ	ラ	シ

x	1	2	4	8	16	32	64
長　さ	全音符	2分音符	4分音符	8分音符	16分音符	32分音符	64分音符

2. タイ(&)

前後の音をつなぎます．前後の音程が異なる場合と音程の後に音程以外の文字を指定すると，前の音だけ "Q8" を指定した場合と同じ効果になります．

3. シャープ(+，#)

音を半音上げます．ただし，"B+(B#)" としても1オクターブ上の "C" にはなりません．同じオクターブの "C" になります．

4. フラット(−)

音を半音下げます．ただし，"C−" としても1オクターブ下の "B" にはなりません．同じオクターブの "B" になります．

5. 符点(．)

音符や休符の長さを1.5倍にします．音の長さを表すパラメータ x の後に指定します．

6. 連符({ } x)

$\frac{\text{指定された長さ x}}{\text{\{ \}の中の音符の数}}$ によって等分された長さで音を発生します．等分した際，1音の長さが64分音符より短くなる場合はエラーになります．また{ }内で音符の長さを変えるコマンドを指定した場合，連符にはなりません．

7. Vコマンド(Vx)

音量の設定をします．パラメータ x は0〜15の範囲の値で設定し，初期値は "V8" です．"V15" が最大音量になります．

8. Lコマンド(Lx)

音の長さを設定します．パラメータ x は1〜64の範囲の値で，初期値は "L4" です．パラメータ x と音符の対応は，音程のパラメータ x と同様です．

9. Qコマンド(Qx)

1音中の音の長さの割り合いを設定します．パラメータ x と長さの割り合いの対応は次のとおりです．

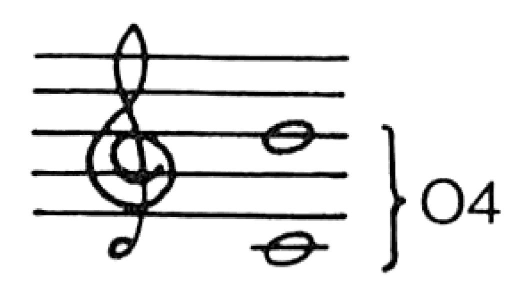
パラメータ	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
音の長さの割合	1/8	2/8	3/8	4/8	5/8	6/8	7/8	8/8

初期値は，1音中の音の長さがすべて出力される "Q8" が指定されます．

MIDIを使用している場合に，"Q8" を指定して同じ高さの音を続けて演奏すると，シンセサイザの機種や音色によって，第2音以降の音が出ないことがあります．そのときはパラメータｘの値を小さく設定してください．

10. Oコマンド(Ox)

オクターブの設定をします．パラメータｘは1～8の値で，初期値は "O4" です．



なお，Oコマンドとは別に現在のオクターブから1オクターブ上げる "＞" と1オクターブ下げる "＜" があります．

11. Rコマンド(Rx)

休符の設定をします．パラメータｘは1～64の値で初期値は "R4" です．パラメータｘと休符の対応は次のとおりです．

x	1	2	4	8	16	32	64
長　さ	全休符	2分休符	4分休符	8分休符	16分休符	32分休符	64分休符

パラメータｘなしのRコマンドを設定した場合，その休符の長さは，Lコマンドが設定されていた場合はその長さと同じになります．また，各チャネルの音を発生する長さが違った場合，長さの短い方の残りの部分はすべてR(レスト)とみなされます．

12. Tコマンド(Tx)

テンポの設定をします．パラメータｘは32～255の値で，初期値は "T120" です．"T120" とは1分間に4分音符が120回演奏できる速さで，楽譜によく書かれている ♩＝120に当たります．

TコマンドとLコマンドのパラメータｘを両方とも大きな値にする(速いテンポで短い音符を演奏するなど)と，時間の計算の誤差により，実際の速さや長さと異なって聞こえる場合があります．

13. Nコマンド(Nx)

パラメータxで指定された高さの音を発生します．パラメータxは0〜96の値です．なお，FM音源の場合，"N96"は"N0"に等しくなります．その他は次のとおりです．

Nコマンドパラメータと音程の対応

	O1	O2	O3	O4	O5	O6	O7	O8	O9
C	0	12	24	36	48	60	72	84	96
C＋(D−)	1	13	25	37	49	61	73	85	
D	2	14	26	38	50	62	74	86	
D＋(E−)	3	15	27	39	51	63	75	87	
E	4	16	28	40	52	64	76	88	
F	5	17	29	41	53	65	77	89	
F＋(G−)	6	18	30	42	54	66	78	90	
G	7	19	31	43	55	67	79	91	
G＋(A−)	8	20	32	44	56	68	80	92	
A	9	21	33	45	57	69	81	93	
A＋(B−)	10	22	34	46	58	70	82	94	
B	11	23	35	47	59	71	83	95	

14. @Wコマンド(@Wx)

パラメータxで指定された長さだけ，現在の状態(Key-on/off)を維持します．Key-onの状態を続けるには，YコマンドでKey-onした後だけ有効です．音程(C〜B)はKey-offしてから次のMMLコマンドに実行を移しますので，Rコマンドと同じ効果しか得られません．パラメータxが省略された場合は，Lコマンドの値が設定されたものとみなします．

15. @コマンド(@x)

FM音源をパラメータxで指定された音色番号の音色に切り替えます．音色番号と音色の対応は音色番号表を参照してください．また，@コマンドを指定した場合はパラメータxを省略することができませんので，必ずパラメータを設定してください．ただし，@コマンドが実行されるまでは音色番号0が指定されているとみなされます．

音色番号表

音色番号	音　色　名	解　　　　　説	音程のずれ
0	Default VOICE	音色番号が全く指定されなかったとき，ハープシコードが割り当てられます．	
1	BRASS 2	ブラス系の音	－8
2	STRING 2	ストリングス系の音	
3	EPIANO 3	エレクトリックピアノ系の音	
4	EBASS 1	エレクトリックベース系の音	－8
5	EORGAN 1	エレクトリックオルガン系の音	
6	PORGAN 1	パイプオルガン系の音	＋8
7	FLUTE	フルート	＋5
8	OBOE	オーボエ	
9	CLARINET	クラリネット	
10	VIBRPHN	ビブラホン	＋5
11	HARPSIC	ハープシコード	
12	BELL	ベル	
13	PIANO	ピアノ	
14	MUSHI	虫の鳴き声	
15	DESCENT	高空から降下する音	
16	UFO	UFOが遠ざかる音	
17	GRANPRI	レーシングカーのエンジン音	
18	LASER 1	レーザーガンの音	
19	LASER 2	レーザーガンの音	
20	SIN WAVE	チューニング用の正弦波	
21	BRASS 1	ブラス系の音	
22	BRASS 2	ブラス系の音，音色番号1と同じ	
23	TRUMPET	トランペット	
24	STRING 1	ストリングス系の音	
25	STRING 2	ストリングス系の音，音色番号2と同じ	
26	EPIANO 1	エレクトリックピアノ系の音	
27	EPIANO 2	エレクトリックピアノ系の音	
28	EPIANO 3	エレクトリックピアノ系の音，音色番号3と同じ	
29	GUITAR	ギター	

音色番号	音　色　名	解　　　　　説	音程のずれ
30	EBASS 1	エレクトリックベース系の音，音色番号4と同じ	－8
31	EBASS 2	エレクトリックベース系の音	－8
32	EORGAN 1	エレクトリックオルガン系の音，音色番号5と同じ	
33	EORGAN 2	エレクトリックオルガン系の音	
34	PORGAN 1	パイプオルガン系の音，音色番号6と同じ	＋8
35	PORGAN 2	パイプオルガン系の音	＋12
36	FLUTE	フルート，音色番号7と同じ	＋5
37	PICCOLO	ピッコロ	＋8
38	OBOE	オーボエ，音色番号8と同じ	
39	CLARINET	クラリネット，音色番号9と同じ	
40	GROCKEN	グロッケン	
41	VIBRPHN	ビブラホン，音色番号10と同じ	＋5
42	XYLOPHN	シロホン	
43	KOTO	琴	
44	ZITAR	ツィター	
45	CLAV	クラビネット	－8
46	HARPSIC	ハープシコード，音色番号0，11と同じ	
47	BELL	ベル，音色番号12と同じ	
48	HARP	ハープ	
49	BELL/BRASS	スタッカートでベル，ロングトーンでブラスの音	
50	HARMONICA	ハーモニカ	
51	STEEL DRUM	スチールドラム	＋5
52	TIMPANI	ティンパニー	／
53	TRAIN	列車の警笛	／
54	AMBULAN	救急車	／
55	TWEET	小鳥のさえずり	／
56	RAIN DROP	雨の落ちる音	／
57	HORN	ホルン	
58	SNARE DRUM	スネアドラム	／
59	COW BELL	カウベル	
60	PERC 1	パーカッション系の音	
61	PERC 2	パーカッション系の音	

＊　音程のずれは，音色番号０のDefault VOICEを基準としてプラス，マイナスで表示してあります．空白の部分はDefault VOICEと同じ音程を持ちます．斜線の部分の音色は音程を持ちません．

音色番号１から12までは，他のPCシリーズとの互換性を保つために用意されています．

16.　@Vコマンド(@Vx)

FM音源とMIDIの音量を細かく設定します．パラメータxは0〜127の値です．なお，このコマンドはVコマンドと関係なく設定されます．"V15"と設定してあっても，"@V0"とすれば音量は０になります．"@V127"が最大音量です．Vコマンドと@Vコマンドのパラメータxの対応は次のようになります．

Vコマンド	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
@Vコマンド	85	87	90	93	95	98	101	103	106	109	111	114	117	119	122	125

@Vコマンドを指定した場合はパラメータxを省略できませんので，必ずパラメータxを指定してください．

17.　Yコマンド(Yr, d)

OPNのレジスタrの内容をdに設定します．OPNレジスタおよび設定値については，**プログラマーズガイド**　第12章５「FM音源の原理/シンセサイザICの構造」を参照してください．

18.　Zコマンド(Zd)

MIDIにデータdを送ります．

19.　Mコマンド(Mx)

SSG音源のエンベロープ周期の設定をします．パラメータxは1〜65535の範囲の値で設定し，初期値は"M255"です．パラメータxは次の式によって求めることができます．

x＝fclock＊T/256　(fclock:　基本周波数(1996800Hz)
T:　周期(秒))

20.　Sコマンド(Sx)

SSG音源のエンベロープ形状の設定をします．パラメータxは0〜15の範囲の値で設定し，初期値は"S1"です．パラメータxとエンベロープ形状の関係は，CMD SOUND文を参照してください．

• MMLの各パラメータには変数を使うことができます．MMLのあとに"＝変数名；"で指定します．

例)　10 VL=15:RG=40:DT=240
　　20 CMD PLAY #2,"V=VL;CDE"
　　30 CMD PLAY #2,"Y=RG;,=DT;@W1"

また，音程のあとに長さを指定する場合にも変数を使うことができます(ただし，ミュージックインタフェースボードに添付のカセットテープの中に入っているプログラムを実行して拡張されたBASICでは使用できません)．

例)　10 LN=2
　　20 CMD PLAY #2,"L4C=LN;DE"

例　CMD PLAY #2,"c","e","g"

• 内蔵のOPNのFM音源で"ドミソ"の和音を出します．なお，この場合"#2,"は省略できます．

プログラム例

```
100 ' CMD PLAY ｻﾝﾌﾟﾙ
110 S=2:T=11
120 CMD PLAY #S,"@=T;T180O514"
130 A$="GB-2O6CD4.E8DC2O5AF4.G8A"
140 B$="O6F2F8.R16F4.E8DC2O5AF4.G8A"
150 CMD PLAY #S,A$+"B-2GG4.F+8GA2F+D2"
160 CMD PLAY #S,A$+"B-4.A8GF+4.E8F+G2G8.R16G2."
170 CMD PLAY #S,B$+"B-2GG4.F+8GA2F+D2."
180 CMD PLAY #S,B$+"B-4.A8GF+4.E8F+G2G8.R16G2."
190 END
Ok
```

• ハープシコードの音色でグリーンスリーブスを演奏します．

注意：• MMLの中でエラーが生じた場合，音色はDefault VOICE(ハープシコード)に戻ります．

• CMD PLAY文で発生させた音程は，A＝440Hzを基準とした音程と多少誤差があります．

• エンベロープに影響するレジスタに与える値によっては音が出なくなることがあります．この場合はCMD STOPM文を実行してください．それでも音が出ない場合は，プログラムをセーブしたあとリセットボタンを押して，もう一度実行しなおしてください．

• BASICの構文解析時間のため，次の行に移動するとき音と音の間にまがあいたり，曲全体の長さが長くなることがあります．

・READ文およびDATA文で音符を読み込んだ場合，ガーベッジコレクションによって音楽が一時的に止まることがありますので，CMD PLAY文でプログラムを作成してください．

・本機にミュージックインタフェースボード(PC-8801-01)を接続して使用する場合，ミュージックインタフェースボードのジャンパスイッチのジャンパコネクタは取り外してください．

CMD SOUND

シー・エム・ディー・サウンド：cmd sound

機能　PSGのレジスタにデータをセットする

書式　CMD SOUND〈レジスタ番号〉,〈数式〉

解説
- このステートメントは，本体にミュージックインタフェースボード(PC-8801-10)が接続してあるときに使用します．
- CMD SOUND文は，ミュージックインタフェースボードに実装されている2個のPSGレジスタに直接データを書き込みます．
- 〈数式〉は0〜255までの整数で指定します．
- 2個のPSGにはそれぞれ16個のレジスタがあり，次のような機能を持っています．

CMD SOUNDレジスタ表

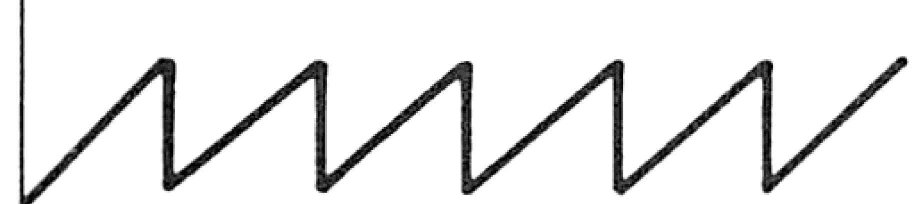
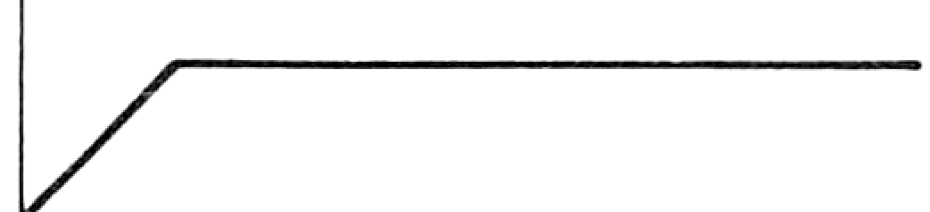
PSG1 レジスタ	PSG2 レジスタ	ビット / 機能	MSB B7	B6	B5	B4	B3	B2	B1	LSB B0
R 0	R16	チャネルA周波数	8BIT FT-A							
R 1	R17						4BIT CT-A			
R 2	R18	チャネルB周波数	8BIT FT-B							
R 3	R19						4BIT CT-B			
R 4	R20	チャネルC周波数	8BIT FT-C							
R 5	R21						4BIT CT-C			
R 6	R22	ノイズ周波数				5BIT NP				
R 7	R23	チャネル設定	$\overline{\text{I/O}}$		$\overline{\text{NOISE}}$			$\overline{\text{TONE}}$		
					C	B	A	C	B	A
R 8	R24	チャネルA音量				M	L3	L2	L1	L0
R 9	R25	チャネルB音量				M	L3	L2	L1	L0
R10	R26	チャネルC音量				M	L3	L2	L1	L0
R11	R27	エンベロープ周期	8BIT FT							
R12	R28		8BIT CT							
R13	R29	エンベロープ形状					E3	E2	E1	E0
R14	R30	I/OポートAデータ	8BITパラレルI/OポートA							
R15	R31	I/OポートBデータ	8BITパラレルI/OポートB							

- FTは微調整，CTは主調整，NPは周波数調整を意味します．
- R7, R23の第6，7ビットおよびR14, R15, R30, R31は，音を出すには直接関係ありません(I/Oポートの機能は無効です)．

• R8, R9, R10, R24, R25, R26は，エンベロープモードにすると音量の設定はできません．

• エンベロープの形状には次のようなものがあります．

エンベロープ形状

R13, R29の値	エンベロープパターン
0, 1, 2, 3, 9	
4, 5, 6, 7, 15	
8	
10	
11	
12	
13	
14	

• 詳しくは，ミュージックインタフェースボードに添付されているユーザーズマニュアルを参照してください．

例

CMD SOUND 7,0

• ミュージックインタフェースボードのPSG1のレジスタ7に0を書き込みます．

プログラム例

• このプログラムを実行する場合には，別売のミュージックインタフェースボードが必要です．

```
100 ' CMD SOUND ｻﾝﾌﾟﾙ
110 CMD STOPM
120 CMD SOUND 7,7
130 CMD SOUND 23,7
140 CMD SOUND 6,30
150 CMD SOUND 22,30
160 CMD SOUND 13,12
170 CMD SOUND 29,12
180 FOR I=1 TO 30
190   FOR L=0 TO 15
200     CMD SOUND 8,L
210   NEXT L
220   FOR L=15 TO 0 STEP -3
230     CMD SOUND 8,L
240   NEXT L
250 NEXT I
260 END
Ok
```

• 蒸気機関車が走る音を出します．

• 120行と130行でノイズの設定を，140行と150行ではノイズの周波数をコントロールしています．

• 160行および170行ではエンベロープの形状を設定しており，200行と230行で音を出しています．

注意： • 本機にミュージックインタフェースボードを接続して使用する場合，ミュージックインタフェースボードのジャンパスイッチのジャンパコネクタは取り外してください．

CMD STOPM

シー・エム・ディー・ストップ・エム：cmd stop music

機能 初期設定を行う

書式 CMD STOPM

解説 ・ボリューム，オクターブ等の音楽機能の初期化を行います．

・CMD PLAY文，CMD SOUND文，CMD VOICE REG文で発生させた音で，Key-on中(鍵盤楽器にたとえれば，鍵盤を押した状態)の音はCMD STOPM文で止めることができます．ただし，Release Rateの設定値によっては音が残るものがあります．

例 CMD STOPM

・初期設定を行います．

プログラム例

```
100 ' CMD STOPM ｻﾝﾌﾟﾙ
110 CMD BGM 0
120 CMD PLAY "@605T80L1"
130 CMD PLAY "V15C&C"
140 FOR W=0 TO 500:NEXT W
150 CMD STOPM
160 CMD PLAY "V15C&C"
170 END
Ok
```

・120行で音色をパイプオルガンにしてオクターブ5，テンポ80，音符の長さ1(全音符)に設定します．そして，130行で音を出します．

・150行でCMD STOPM文を実行すると，160行ではオクターブ4，テンポ120，音符の長さ4(4分音符)のパイプオルガンの音で演奏されます．

注意：・本機にミュージックインタフェースボード(PC-8801-10)を接続して使用する場合，ミュージックインタフェースボードのジャンパスイッチのジャンパコネクタは取り外してください．

CMD UNLINK

シー・エム・ディー・アンリンク：cmd unlink

機能 拡張命令を解除する

書式 CMD UNLINK

解説
- 拡張命令を解除し，専有していたエリアをフリーエリアとして開放します.
- このステートメントを実行すると，CLEAR, &HE5FFが自動的に行われます.
- CMD UNLINK文を実行すると，CMDステートメントは使えなくなります.

例 CMD UNLINK

- 拡張命令を解除します.

実行例

```
cmd unlink⏎
Ok
```

- 拡張命令を解除し，専有していたエリアをフリーエリアにします.

CMD VOICE

シー・エム・ディー・ボイス：cmd voice

機能 音色を設定する

書式 CMD VOICE 〈整数配列名1〉〔,〈整数配列名2〉〕〔,〈整数配列名3〉〕

解説
- 指定された配列変数の値を音色設定のパラメータとしてFM音源に送り，音色を設定します．
- 整数配列名1～3は，それぞれチャネル1～3に対応します．
- 配列変数はDIM文で定義し，添字は(4, 9)で宣言します．
- 各配列変数の意味は次のようになります．ここでは配列変数名をV%として説明しています．

V%(0, 0): Feedback(フィードバック)/Algorithm(アルゴリズム)

これは6ビットのデータになっていて，以下の意味を持っています．

0	0	D5	D4	D3	D2	D1	D0
		Feedback			Algorithm		

使用例) V%(0,0)＝8＊FB＋AR

FBはFeedbackを意味し，0～7の値を指定します．

ARはAlgorithmを意味し，0～7の値を指定します．

Feedback

第1オペレータの変調度を決めます．指定できる範囲は0～7までで，大きく指定するほど金属的な音になります．

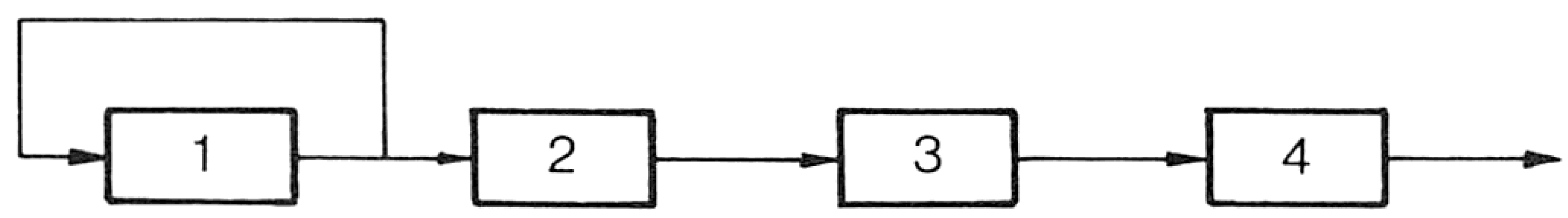
第1オペレータは自分自身の出力を変調信号としているため，この変調度をこれにより制御します．

値	0	1	2	3	4	5	6	7
変調度	OFF	π/16	π/8	π/4	π/2	π	2π	4π

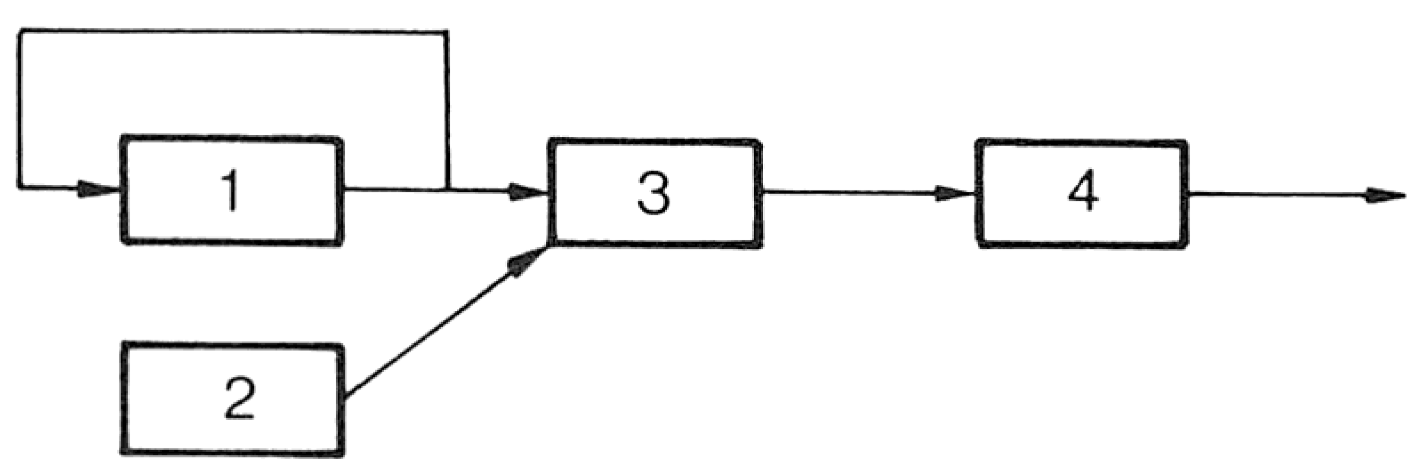
Algorithm

4つのオペレータの接続の仕方を決定します．指定できる範囲は0～7までで，それぞれ次のようなアルゴリズムを選択します．

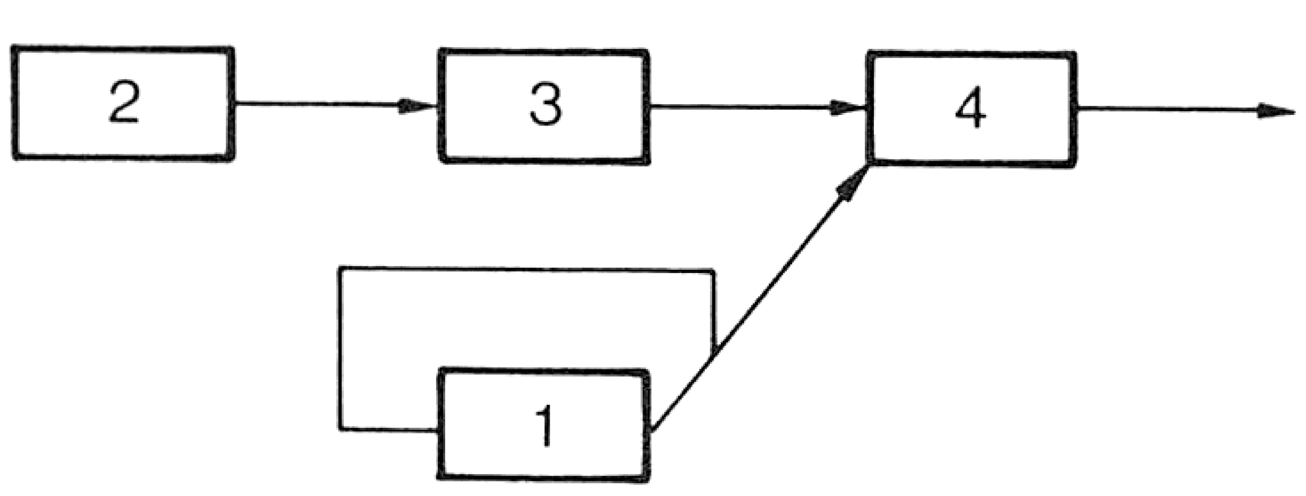
0 :



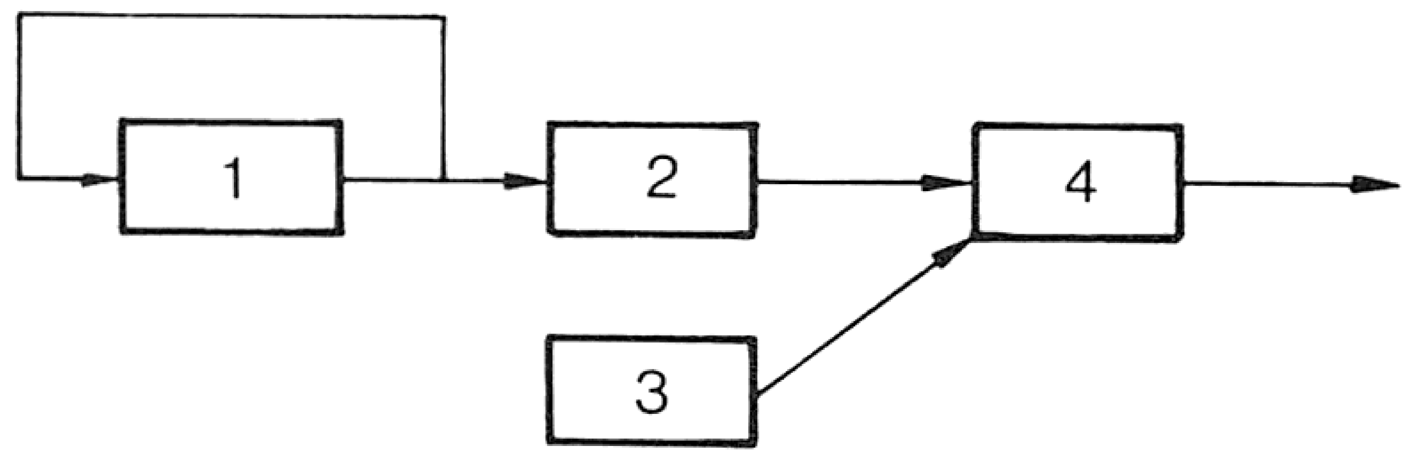
1 :



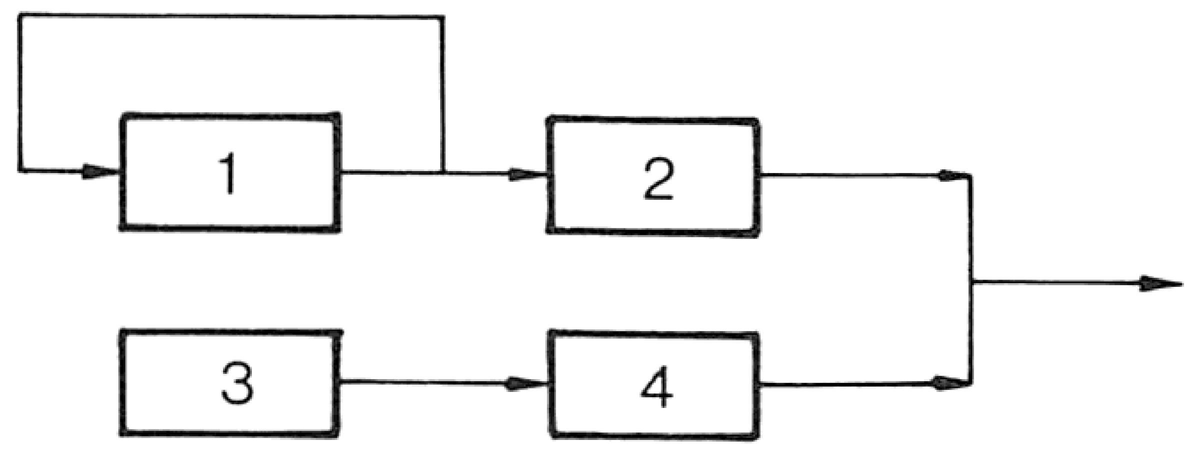
2 :



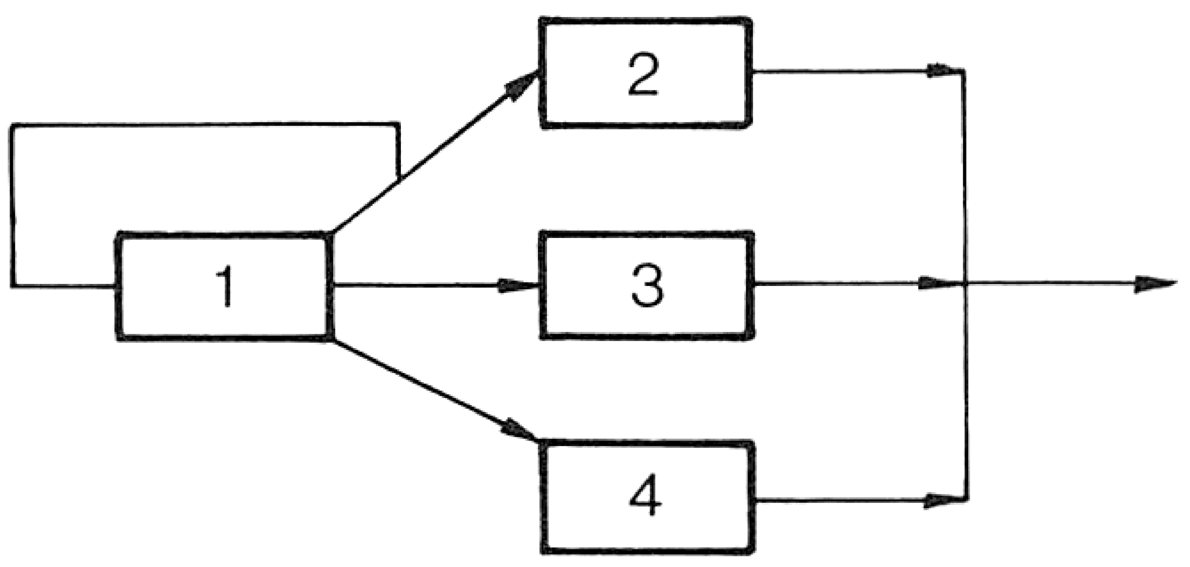
3 :



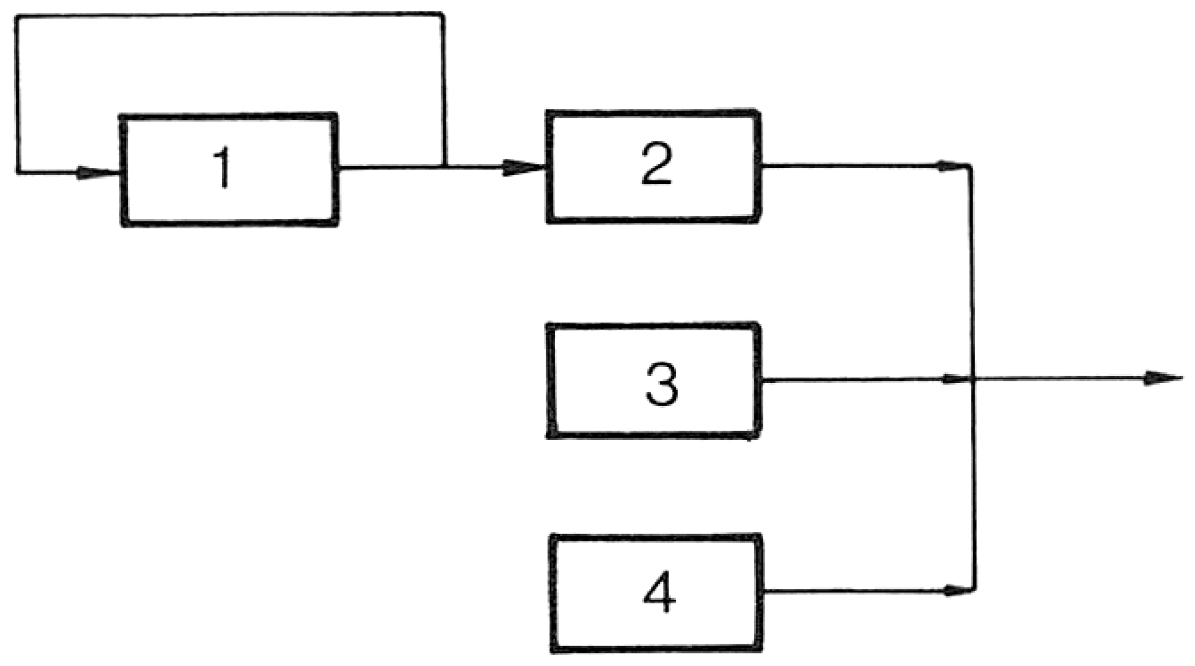
4 :



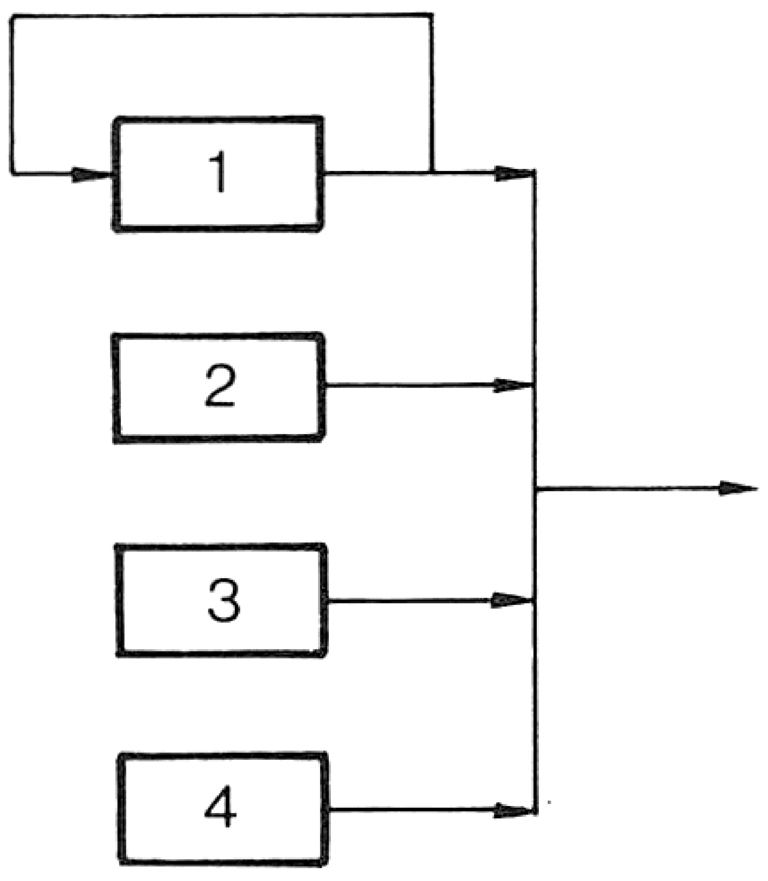
5 :



6 :



7 :

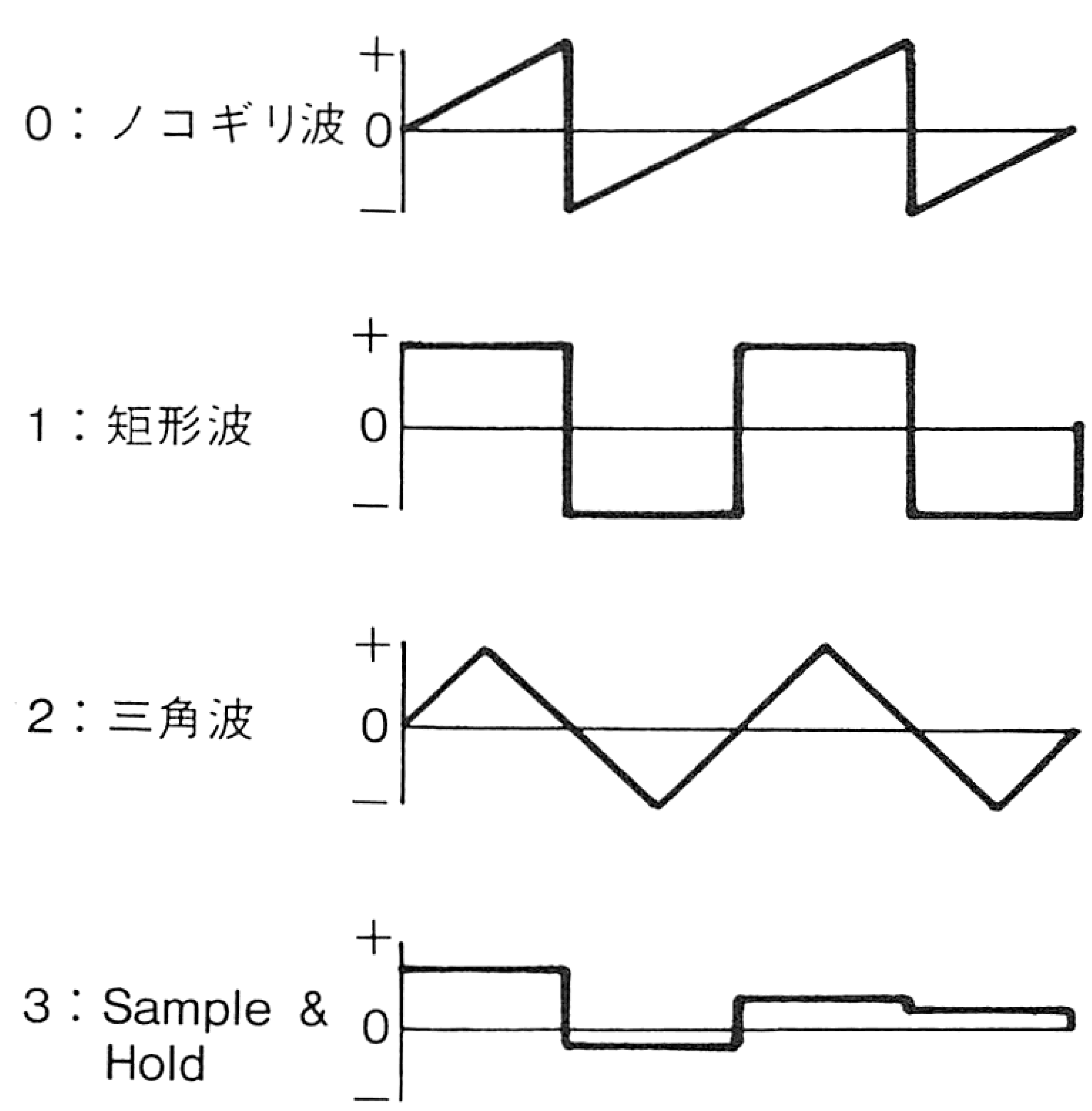


V%(0, 1):　Operator mask(オペレータマスク)

各オペレータを使用するか否かを決定します．0でオペレータをOFF，1でオペレータをONにします．それぞれの値は次のとおりです．

オペレータ				値
OP4	OP3	OP2	OP1	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

V%(0, 2):　Wave Form(ウェーブフォーム)



Sample & Hold(サンプルアンドホールド)はランダム値が出力されます．

V%(0, 3):　Sync(シンク)

0: Key-on(鍵盤楽器にたとえれば，鍵盤を押した状態)と無関係にLFO(Low Frequency Oscillator)が動作します.

1: Key-onと同期してLFOが動作します.

V%(0, 4):　Speed(スピード)

LFOの速度を決めます. 指定できる範囲は0～16383までで，大きいほど速くなります.

V%(0, 5):　Pitch Modulation Depth(ピッチモジュレーションデプス)

音程に対してLFOをかける深さを決めます. 指定できる範囲は－127～127までで，絶対値が大きいほど深くかかります. 負の値をとると，波形の極性が逆になります.

V%(0, 6):　Amplitude Modulation Depth(アンプリチュードモジュレーションデプス)

音量に対してLFOをかける深さを決めます. 指定できる範囲は－127～127までで，絶対値が大きいほど深くかかります. 負の値をとると波形の極性が逆になります.

V%(0, 7):　Pitch Modulation Sensitivity(ピッチモジュレーションセンシティビティ)

音程に対するLFOをかける度合いを決定します. 指定できる範囲は0～15までで，数値が大きいほど変化の幅が大きくなります. 通常，Pitch Modulation Sensitivityで大まかな調整をしておいて，Pitch Modulation Depthで細かく調整します.

V%(0, 8):　未使用

V%(0, 9):　未使用

• OPはオペレータ番号を意味します. 1～4の値を指定します.

V%(OP, 0):　Attack Rate(アタックレート)

音が発生(Key-on)してからEG(Envelop Generator)の出力レベルが最大になるまでの時間を決めます. 指定できる範囲は0～31までで，数値が大きいほど短くなります.

V%(OP, 1):　Decay Rate(ディケイレート)

EGの出力レベルが最大になってからSustain Levelに達するまでの時間を決めます. 指定できる範囲は0～31までで，数値が大きいほど短くなります.

V%(OP, 2): Sustain Rate（サスティンレート）

Sustain LevelからEGの出力レベルが0になるまでの時間を決めます．指定できる範囲は0～31までで，数値が大きいほど短くなります．

V%(OP, 3): Release Rate（リリースレート）

Keyを離してからEGの出力レベルが0になるまでの時間を決めます．指定できる範囲は0～15までで，数値が大きいほど短くなります．

V%(OP, 4): Sustain Level（サスティンレベル）

DecayからSustainに移るときのレベル（減衰量）を決めます．指定できる範囲は0～15までで，数値が大きいほどEGの出力レベルが低くなります．

V%(OP, 5): Output Level（アウトプットレベル）

これは減衰量のため，0が最大出力になります．指定できる範囲は127～0までです．

V%(OP, 6): Keyboard Rate Scaling Depth（キーボードレートスケーリングデプス）

高音になるほどEGの変化速度を速くする度合いを決めます．指定できる範囲は0～3までです．

V%(OP, 7): Multiple（マルチプル）

周波数比を決定します．指定できる範囲は0～15までで，0で½になります．

V%(OP, 8): Detune（デチューン）

音程のずれの度合いを決めます．指定できる範囲は−3～3までです．

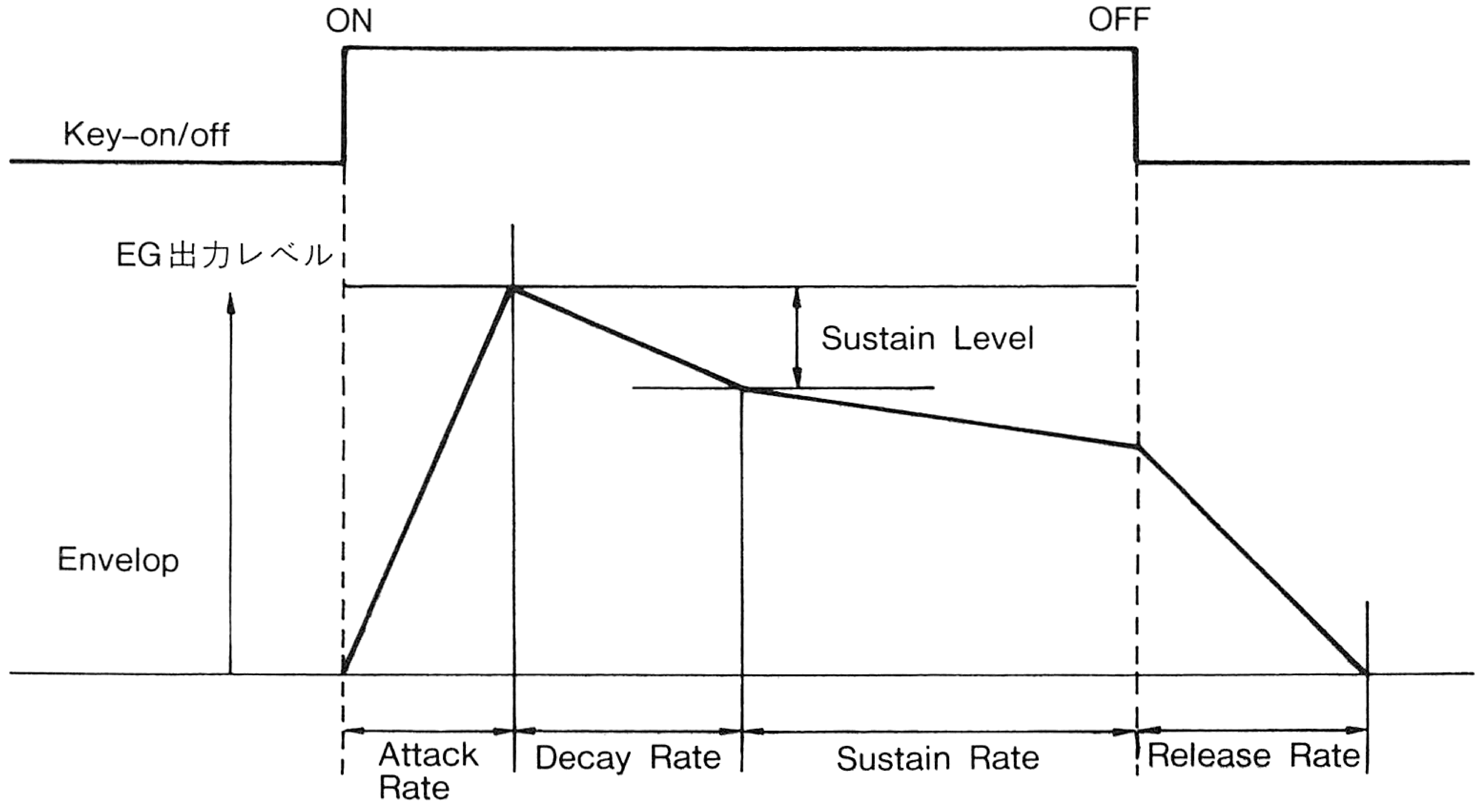
V%(OP, 9): Amplitude Modulation Sensitivity（アンプリチュードモジュレーションセンシティビティ）

音量に対するLFOをかける度合いを決定します．指定できる範囲は0～15までで，数値が大きいほど変化の幅が大きくなります．通常，Amplitude Modulation Sensitivityで大まかな調整をしておいて，Amplitude Modulation Depthで細かく調整します．

- EG（Envelop Generator…エンベロープジェネレータ）

楽器の音は時間とともに音量，音色が変化します．この時間的な変化をエンベロープといいます．そして，このエンベロープを人工的に作る機能をEG（Envelop Generator）といいます．

EGは以下のパラメータを持っており，それを設定することによって，より自然の楽器に近い音色を付けることができます．



例　CMD VOICE A%, B%, C%

• 整数配列変数A%，B%，C%に入っている音色データを各チャネルに設定します．

プログラム例

```
100 ' CMD VOICE ｻﾝﾌﾟﾙ
110 DIM A%(4,9)
120 CMD BGM Ø
130 CMD VOICE COPY 2Ø,A%
140 CMD VOICE A%
150 CMD PLAY "O6V15L1C&C&C&C"
160 FOR W=Ø TO 1ØØØ:NEXT W
170 A%(Ø,2)=1
180 A%(Ø,3)=1
190 A%(Ø,4)=45Ø
200 A%(Ø,5)=1ØØ
210 A%(Ø,7)=8
220 CMD VOICE A%
230 CMD PLAY "O6V15L1C&C&C&C"
240 END
Ok
```

• 2種類の音を出します．

• 130行で配列A%に音色番号20(正弦波)のデータをコピーします．

• 140行で，音色番号20のデータをそのままFM音源に送り，150行で音を出します．

• 170行から210行でデータを多少書き換えます．

• 220行で，変更したデータをFM音源に送り，230行で音を出します．

参照：プログラマーズガイド　第12章　「サウンド機能」

CMD VOICE COPY

シー・エム・ディー・ボイス・コピー：**cmd voice copy**

機能 音色データを配列変数にコピーする

書式 CMD VOICE COPY〈音色番号〉,〈整数配列名〉

解説
- 〈音色番号〉で示されたFM音源のパラメータの値を，〈整数配列名〉で指定された配列変数にコピーします．この配列変数は，CMD VOICE文で新しく音色を設定するために用います．
- 配列変数はDIM文で定義し，添字は(4, 9)で宣言します．
- 〈音色番号〉についてはCMD PLAY文を参照してください．

例 CMD VOICE COPY 7,F%
- フルートの音色データを整数配列変数F%にコピーします．

プログラム例

```
100 ' CMD VOICE COPY ｻﾝﾌﾟﾙ
110 DIM A%(4,9)
120 CMD BGM 0
130 CMD VOICE COPY 20,A%
140 CMD VOICE A%
150 CMD PLAY "O6V15L1C&C&C&C"
160 FOR W=0 TO 1000:NEXT W
170 A%(0,2)=1
180 A%(0,3)=1
190 A%(0,4)=450
200 A%(0,5)=100
210 A%(0,7)=8
220 CMD VOICE A%
230 CMD PLAY "O6V15L1C&C&C&C"
240 END
Ok
```

- 2種類の音を出します．
- このプログラム例はCMD VOICE文と同じものです．

参照：DIM, CMD PLAY, CMD VOICE

CMD VOICE LFO

シー・エム・ディー・ボイス・エル・エフ・オー：cmd voice lfo

機能 各チャネルにLFO効果を与える

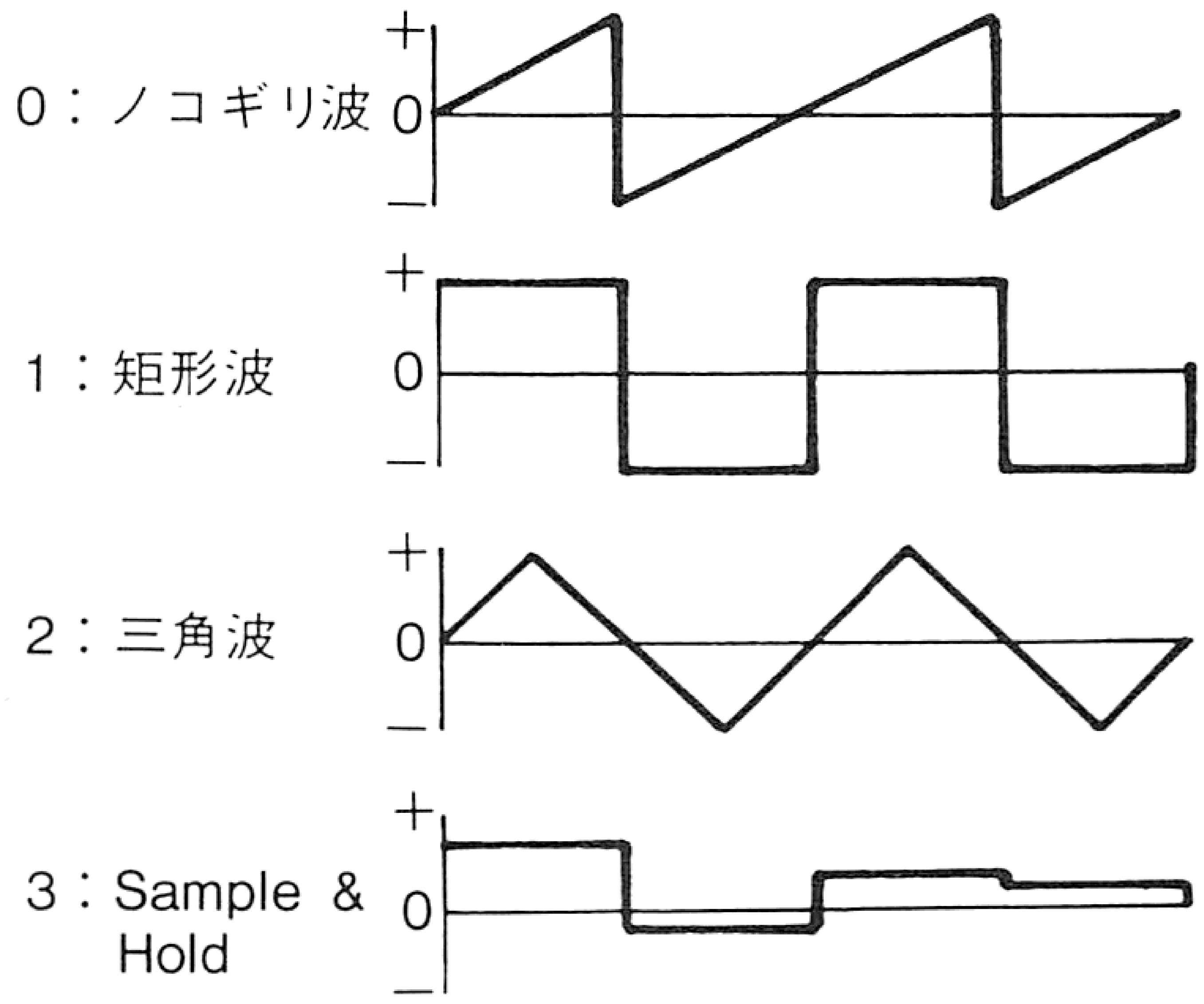
書式 CMD VOICE LFO 〈チャネル番号〉〔，〈wf〉〕〔，〈syncスイッチ〉〕〔，〈スピード〉〕〔，〈pmd〉〕〔，〈amd〉〕〔，〈pms〉〕

解説 • 指定したチャネルの音にビブラート（音程を細かく変化させること）やトレモロ（音量を細かく変化させること）を付けます．

• ビブラートやトレモロの効果の度合いは，LFO(Low Frequency Oscillator)と呼ばれる低周波発振器のパラメータによって制御します．パラメータは次のように設定します．

〈チャネル番号〉…1～6の値を指定します．この値により，LFO効果を付けるチャネルを設定します．

〈wf〉………………Wave Form．波形によりLFOの変化の仕方を設定します．0～3の値を指定することにより，次のような波形を選ぶことができます．



Sample & Hold(サンプルアンドホールド)はランダム値が出力されます．

〈syncスイッチ〉…synchroスイッチが1の場合，音を発生させると同時にLFO波形を発生させます．0の場合，音の発生と無関係にLFO波形を発生させます．

〈スピード〉………0～16383の値を指定することにより，LFOの周波数を決定します．数が大きいほど周波数が高くなります．

〈pmd〉……………Pitch Modulation Depth．音程に対してLFOをかける深さを決定します．つまり，ビブラートをかける場合に使います．指定できる値は−127～127までで，絶対値が大きいほど変化の幅が大きくなります．負の値を指定すると，LFO波形の極性が逆転します．

〈amd〉……………Amplitude Modulation Depth．音量に対してLFOをかける深さを決定します．つまり，トレモロをかける場合に使います．ただし，〈amd〉が設定できるのはFM音源のときだけです．指定できる値は－127～127までで，絶対値が大きいほど変化の幅が大きくなります．負の値を指定すると，LFO波形の極性が逆転します．

〈pms〉……………Pitch Modulation Sensitivity．〈pmd〉と同様，音程に対してLFOをかける度合いを決定します．指定できる値は0～15までで，数値が大きくなるほど変化の幅が大きくなります．通常，〈pms〉で大まかな調整をしておいて，〈pmd〉で細かい調整をします．

例　CMD VOICE LFO 1,2,1,100,90

• チャネル番号1の音にLFO効果を設定します．

プログラム例

```
100 ' CMD VOICE LFO ｻﾝﾌﾟﾙ
110 CMD BGM 0
120 CMD PLAY "@20V15O6L1"
130 CMD PLAY "C&C&C&C"
140 FOR W=0 TO 1000:NEXT W
150 CMD VOICE LFO 1,3,1,10000,100,,15
160 CMD PLAY "C&C&C&C"
170 END
Ok
```

• 正弦波の音にLFO効果を設定して，ランダムな音を出します．

• 150行で，チャネル1にビブラートをかけるために，wf=3，スピード=10000，pmd=100，pms=15に設定します．

注意： • チャネル1～6のすべてにLFOをかけると，多少テンポが狂う場合があります．

CMD VOICE REG

シー・エム・ディー・ボイス・レジ：cmd voice register

機能 シンセサイザIC(OPN)のレジスタに値を設定する

書式 CMD VOICE REG 〈OPNレジスタ番号〉,〈数式〉

解説
- 指定されたOPNのレジスタに，〈数式〉によって指定された値を代入します.
- 〈数式〉の値の範囲は 0～255です.
- OPN レジスタについては，**プログラマーズガイド** 第12章 5 「FM音源の原理/シンセサイザICの構造」を参照してください.

例 CMD VOICE REG &HA0,1

- OPNのレジスタ(A0H)にデータ(1)を設定します.

プログラム例

```
100 ' CMD VOICE REG ｻﾝﾌﾟﾙ
110 CMD STOPM
120 CMD VOICE REG 7,7
130 CMD VOICE REG 6,30
140 CMD VOICE REG 13,12
150 FOR I=1 TO 30
160   FOR L=0 TO 15
170     CMD VOICE REG 8,L
180   NEXT L
190   FOR L=15 TO 0 STEP -3
200     CMD VOICE REG 8,L
210   NEXT L
220   IF I MOD 15=0 THEN CMD PLAY "@53V15O5CC2"
230 NEXT I
240 END
Ok
```

- 蒸気機関車が走っている音を出します.
- 120行で，SSG音源の 3 音がすべてノイズを出力するように設定します.
- 130行と140行で，ノイズの発生周期およびエンベロープタイプを設定します.
- 150行から230行で音量の調節を行います.

資　料

資料1　N_{88}-BASIC/N_{88}-日本語BASICの予約語

ABS	DATA	GOTO	LPOS	PSET	TAN
AKCNV$**	DATE$	HELP	LPRINT	PUT	TERM
AND	DEF	HEX$	LSET	RANDOMIZE	THEN
ASC	DEFDBL	IEEE	MAP	RBYTE	TIME$
ATN	DEFINT	IF	MERGE	READ	TO
ATTR$	DEFSNG	IMP	MID$	REM	TROFF
AUTO	DEFSTR	INKEY$	MKD$	RENUM	TRON
BEEP	DELETE	INP	MKI$	RESTORE	USING
BLOAD	DIM	INPUT	MKS$	RESUME	USR
BSAVE	DSKF	INSTR	MOD	RETURN	VAL
CALL	DSKI$	INT	MON	RIGHT$	VARPTR
CDBL	DSKO$	IRESET	MOTOR	RND	VIEW
CHAIN	EDIT	ISET	NAME	ROLL	WAIT
CHR$	ELSE	KACNV$**	NEW	RSET	WBYTE
CINT	END	KANJI	NEXT	RUN	WEND
CIRCLE	EOF	KEY	NOT	SAVE	WHILE
CLEAR	EQV	KILL	OCT$	SCREEN	WIDTH
CLOSE	ERASE	KLEN**	OFF	SEARCH	WINDOW
CLS	ERL	KPLOAD**	ON	SET	WRITE
CMD	ERR	KPOS**	OPEN	SGN	XOR
COLOR	ERROR	LEFT$	OPTION	SIN	+
COM	EXP	LEN	OR	SPACE$	−
COMMON	FIELD	LET	OUT	SPC(	*
CONSOLE	FILES	LFILES	PAINT	SQR	/
CONT	FIX	LINE	PEEK	SRQ	^
COPY	FN	LIST	PEN	STATUS	¥
COS	FOR	LLIST	POINT	STEP	'
CSNG	FPOS	LOAD	POKE	STOP	>
CSRLIN	FRE	LOC	POLL	STR$	=
CVD	GET	LOCATE	POS	STRING$	<
CVI	GO TO	LOF	PRESET	SWAP	
CVS	GOSUB	LOG	PRINT	TAB(	

（IEEE, IRESET, SET, POLL, RBYTE, SRQ, STATUS, WBYTEなどは拡張命令に用意されていますので標準では使えません．）

＊＊印のついている予約語はN_{88}-日本語BASICだけに用意されます．

資料２　誘導関数

BASICが"組み込み"関数として用意していない三角関数のうち，いくつかは，用意された関数を使って作り出すことができます．以下の数式を参考にしてください(誤差の範囲に注意が必要です)．

目的とする関数	組み込み関数からの誘導式
セカント	SEC(X)=1/COS(X)
コセカント	CSC(X)=1/SIN(X)
コタンジェント	COT(X)=1/TAN(X)
アークサイン	ARCSIN(X)=ATN(X/SQR(−X*X+1))
アークコサイン	ARCCOS(X)=−ATN(X/SQR(−X*X+1))+1.5708
アークセカント	ARCSEC(X)=ATN(SQR(X*X−1))+(SGN(X)−1)*1.5708
アークコセカント	ARCCSC(X)=ATN(1/SQR(X*X−1))+(SGN(X)−1)*1.5708
アークコタンジェント	ARCCOT=−ATN(X)+1.5708
ハイパーボリックサイン	SINH(X)=(EXP(X)−EXP(−X))/2
ハイパーボリックコサイン	COSH(X)=(EXP(X)+EXP(−X))/2
ハイパーボリックコタンジェント	TANH(X)=−EXP(−X)/(EXP(X)+EXP(−X))*2+1
ハイパーボリックセカント	SECH(X)=2/(EXP(X)+EXP(−X))
ハイパーボリックコセカント	CSCH(X)=2/(EXP(X)−EXP(−X))
ハイパーボリックコタンジェント	COTH(X)=EXP(−X)/(EXP(X)−EXP(−X))*2+1
ハイパーボリックアークサイン	ARCSINH(X)=LOG(X+SQR(X*X+1))
ハイパーボリックアークコサイン	ARCCOSH(X)=LOG(X+SQR(X*X−1))
ハイパーボリックアークタンジェント	ARCTANH(X)=LOG((1+X)/(1−X))/2
ハイパーボリックアークセカント	ARCSECH(X)=LOG((SQR(−X*X+1)+1)/X)
ハイパーボリックアークコセカント	ARCCSCH(X)=LOG((SGN(X)*SQR(X*X+1)+1)/X)
ハイパーボリックアークコタンジェント	ARCCOTH(X)=LOG((X+1)/(X−1))/2
常用対数	LOG_{10}(X)=LOG(X)/LOG(10)

資料３　コントロールコード

16進	10進	キー入力	動　　作
01H	1	コントロール ＋ A	ヘルプキーと同じ.
02H	2	コントロール ＋ B	１つ前のワードに戻る.
03H	3	コントロール ＋ C	実行の中断(停止 と同じ).
04H	4	コントロール ＋ D	カーソル位置から１ワードを削除.
05H	5	コントロール ＋ E	カーソル位置から，その行の終わりまでを消去.
06H	6	コントロール ＋ F	１つ先のワードへ進む.
07H	7	コントロール ＋ G	ブザーを鳴らす.
08H	8	コントロール ＋ H	１文字を削除(削除 と同じ).
09H	9	コントロール ＋ I	タブ位置までカーソルを移動(タブ位置は８桁に設定されている). (タブ と同じ)
0AH	10	コントロール ＋ J	ラインフィード，インサートモードで２行に分割.
0BH	11	コントロール ＋ K	カーソルをホームポジション(左上スミ)に移動(シフト ＋ 画面消去 と同じ).
0CH	12	コントロール ＋ L	テキスト画面を消去する(画面消去 と同じ).
0DH	13	コントロール ＋ M	キャリッジリターン(↵ と同じ).
0FH	15	コントロール ＋ O	プログラム実行中に画面の表示を無効にする.
12H	18	コントロール ＋ R	インサートモードにする(挿入 と同じ).
13H	19	コントロール ＋ S	実行を一時停止する.
15H	21	コントロール ＋ U	１行キャンセル
18H	24	コントロール ＋ X	カーソルを行の最後に移す.
1CH	28	→	カーソルを右へ移動.
1DH	29	←	カーソルを左へ移動
1EH	30	↑	カーソルを上へ移動.
1FH	31	↓	カーソルを下へ移動.

16進	10進	動　　　　　作
07H	7	ブザーを鳴らします.
09H	9	タブ設定までカーソルを移動させます. タブ位置は8桁ごとに設定されています.
0AH	10	カーソルを1つ下の行へ移動させます.
0BH	11	カーソルをホームポジション(左上スミ)に移動させます.
0CH	12	テキスト画面を消去します.
0DH	13	カーソルを行の左端へ移動させます.
16H	22	半角文字入力モードにします.(注)
17H	23	全角文字入力モードにします.(注)
1CH	28	カーソルを1つ右へ移動させます.
1DH	29	カーソルを1つ左へ移動させます.
1EH	30	カーソルを上の行へ移動させます.
1FH	31	カーソルを下の行へ移動させます.

(注) N_{88}-日本語BASICでのみ有効です.

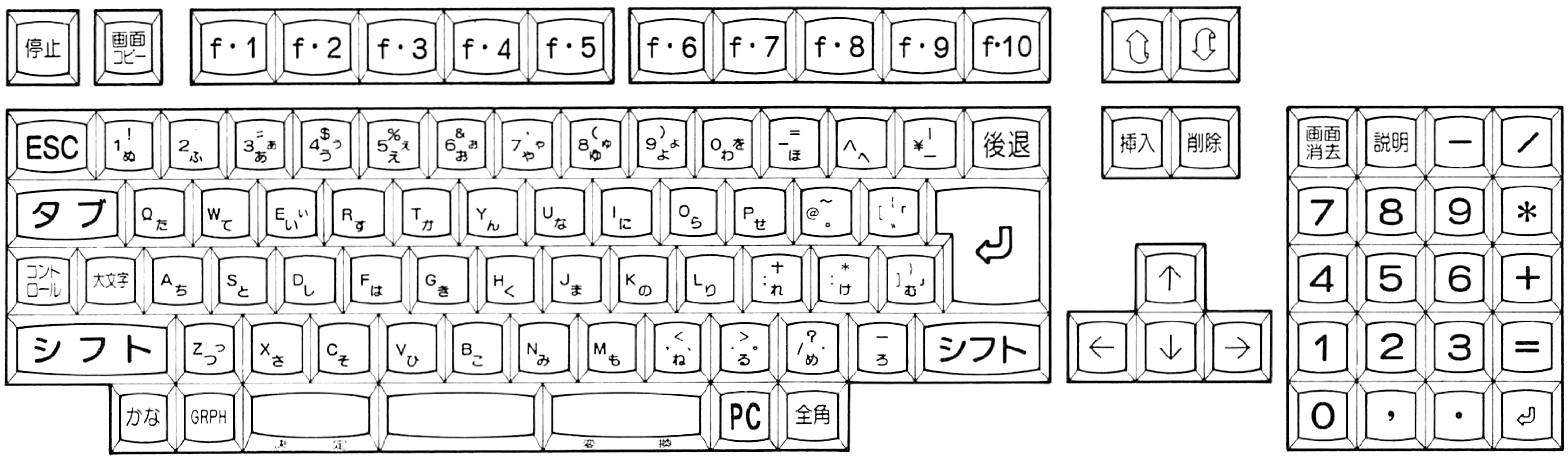
資料4　キャラクタコード

上位4ビット→

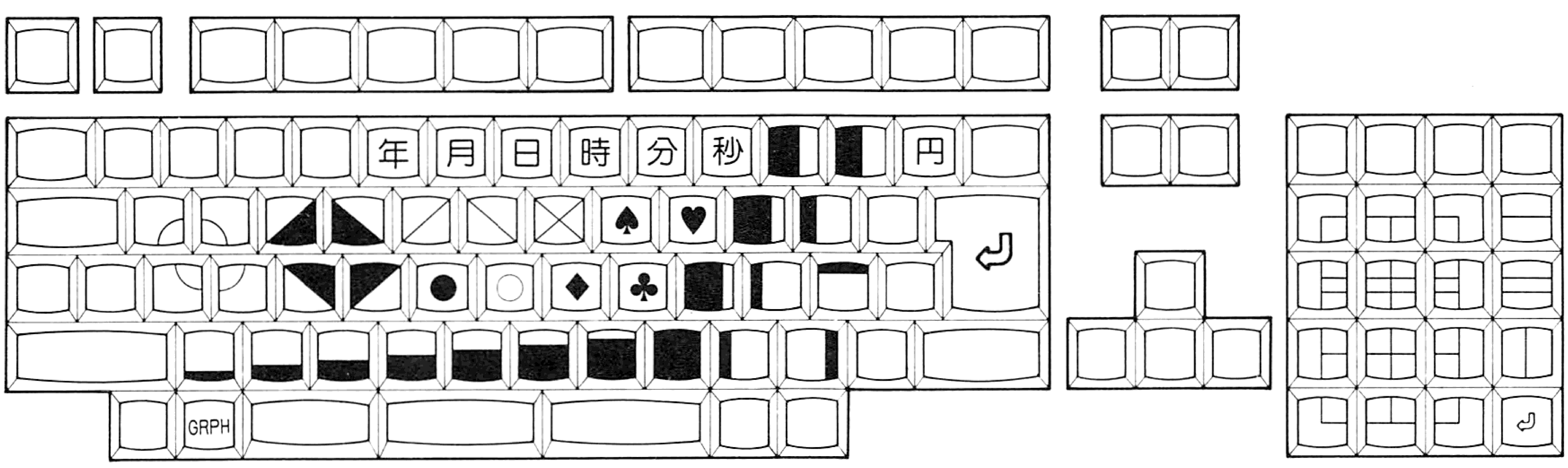
下位4ビット↓

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		D_E		0	@	P		p				ー	タ	ミ		
1	S_H	D_1	!	1	A	Q	a	q			。	ア	チ	ム		円
2	S_X	D_2	"	2	B	R	b	r			「	イ	ツ	メ		年
3	E_X	D_3	#	3	C	S	c	s			」	ウ	テ	モ		月
4	E_T	D_4	$	4	D	T	d	t			、	エ	ト	ヤ		日
5	E_Q	N_K	%	5	E	U	e	u			・	オ	ナ	ユ		時
6	A_K	S_N	&	6	F	V	f	v			ヲ	カ	ニ	ヨ		分
7	B_L	E_B	'	7	G	W	g	w			ァ	キ	ヌ	ラ		秒
8	B_S	C_N	(	8	H	X	h	x			ィ	ク	ネ	リ	♠	
9	H_T	E_M	)	9	I	Y	i	y			ゥ	ケ	ノ	ル	♥	
A	L_F	S_B	*	:	J	Z	j	z			ェ	コ	ハ	レ	♦	
B	H_M	E_C	+	;	K	[	k	{			ォ	サ	ヒ	ロ	♣	
C	C_L	→	,	<	L	¥	l	¦			ャ	シ	フ	ワ	●	
D	C_R	←	-	=	M	]	m	}			ュ	ス	ヘ	ン	○	
E	S_O	↑	.	>	N	^	n	~			ョ	セ	ホ	゛		
F	S_I	↓	/	?	O	_	o				ッ	ソ	マ	゜		

資料5　キーボードレイアウト



キーの配列



グラフィックシンボルキーの配列

資料6　エラーメッセージ

N_{88}-BASIC, N_{88}-日本語BASICは，プログラムの実行を中断させなければならないようなエラーを実行時に検出したとき，エラーメッセージをプリントし，コマンドレベルに戻ります．

ダイレクトモードでのエラーメッセージの書式は，

XX

プログラムモードでの書式は，

XX in lllll

XXはエラーメッセージで，lllllはエラーが検出された行番号です．

ただし，N_{88}-BASICあるいはN_{88}-日本語BASICを使用していて，以下のようなメッセージが表示される場合があります．これはエラーメッセージではありませんので，ON ERROR GOTO文によるエラー処理は行われません．

(1)　?Redo from start

INPUT文で指定した変数の型，あるいは個数がキーボードから入力したデータの型，または個数と一致していないことを表します．

(2)　1 copies of allocation bad on drive〈ドライブ番号〉，または，

2 copies of allocation bad on drive〈ドライブ番号〉

このメッセージはフロッピーディスクを使用している場合，そのフロッピーディスクの内容が壊れかかっているときに表示されます．このような状態になったときは，そのフロッピーディスクのコピーをとることをおすすめします．

資料6.1　N_{88}-BASICおよびN_{88}-日本語BASICで表示されるエラーメッセージ

N_{88}-BASICおよびN_{88}-日本語BASICで表示されるエラーメッセージをアルファベット順に表記し，エラーメッセージの意味，エラーが起こった原因とその解説をします．

コード番号はエラー別に決まっていて，そのエラーが起こったときERR関数に代入されます．[ ]内の数がコード番号を表しています．

Bad allocation table　[69]

意味　フロッピーディスク上のFAT(フロッピーディスクの中のファイル管理をしている部分)が壊れている．

原因　• フロッピーディスクが入っていないドライブに対して，ロードやセーブを行った(ミニフロッピーディスクユニット使用時のみ)．

• フロッピーディスクドライブのレバーが閉まっていない(ミニフロッピーディスクユニット使用時のみ).

• フロッピーディスクに対しOPEN文を実行し，ファイルを書き込んだあとCLOSE文を実行しなかった.

• DSKO$文や機械語モニタ(MON)の [コントロール] + [W] コマンドで，FATを書きなおしてしまった.

解説 ミニフロッピーディスクを使用しているとき，フロッピーディスクを入れずに，あるいはドライブのふたを閉めずにフロッピーディスクをアクセスすると，ドライブのアクセス表示用LEDランプがつきっぱなしになり，しばらくすると上記のエラーメッセージが表示されます．

また，OPEN文を実行したあとにCLOSE文を実行しなかったり，DSKO$文や機械語モニタでFATを書きなおしてしまった場合，そのフロッピーディスクに対してファイルの入出力は不可能になります．

Bad drive number [70]

意味 フロッピーディスクのドライブ番号が誤っている．

原因 システムに接続されていないドライブ番号を指定した．たとえば，2 ドライブしかフロッピーディスクドライブのないシステムで，ドライブ番号に3を指定した．

解説 システムの起動時に，指定可能なドライブ番号が決定されます．それより大きい番号を指定したときにこのメッセージが表示されます．

Bad file name [56]

意味 ファイル名の指定の仕方が間違っている．

原因 LOAD, SAVE, KILL, NAME, OPEN, RUNなどの命令でファイル名を指定するとき，ファイル名に使えない文字や記号がある．

解説 ファイル名の中にキャラクタコード0～31(0H～1F)の文字を含むことはできません．また，ファイル名の最初の1文字にキャラクタコード255(FFH)の文字を使用することはできません．

Bad file number [52]

意味 使用できないファイル番号を使おうとした．

原因 起動時に指定したファイル番号より上のファイル番号をOPEN, PRINT #, WRITE #, GET, PUT等で使っている．

解説 ファイル番号は，「How many files(0-15)?」が表示されたときに入力したファイル数しか使用できません．指定できるファイル番号の範囲は1から15までです．

Bad track/sector [71]

意味 サーフェス，トラック，セクタ番号の指定が誤っている．

原因 DSKI$関数, DSKO$命令のパラメータの中のサーフェス番号，トラック番号，セクタ番号が指定できる値の範囲にない．

解説　DSKI$やDSKO$で直接フロッピーディスクに対して入出力を行うとき，ディスクドライブやシステムディスクの種類によって指定できるパラメータの範囲は異なります．この値は，DSKF関数を使って調べることができます．

Can't continue　[17]

意味　CONT命令でプログラムの続行ができない．

原因　• プログラムをストップさせてから編集をすると，CONT命令によるプログラムの再開・続行ができない．

• プログラム中にステートメントとしてCONTが書いてある．

解説　プログラムをストップさせて変数の値を調べたり，変数に値を代入したり，LISTを取ったりしてもCONT命令による実行の再開は可能ですが，プログラムを一部でも書き換えたり，CLEAR文を実行したり，画面に表示したプログラム上にカーソルを移動して[⏎]を押しただけでもこのエラーの原因になります．

Deleted record　[72]

意味　消去済みのレコードをアクセスしようとした(本機では，このエラーは発生しません)．

Direct statement in file　[57]

意味　アスキー形式のBASICプログラムファイル中に行番号のない行が存在した．

原因　• SAVE命令でアスキーセーブしたBASICプログラム以外のアスキー形式ファイルをロードした．

• SAVE命令でアスキーセーブしたBASICプログラムファイルの内容が，何らかの原因で破壊されているのに，そのファイルをロードしようとした．

• RS-232CポートよりBASICプログラムをロードしているときに，行番号のない行が存在した．

解説　BASICプログラムの各行は，必ず行番号ではじまりますが，アスキー形式のファイルをディスクやRS-232Cポートからロードするときには，どんなデータが読み込まれるか予測できません．

したがって，BASICでは各行の先頭に行番号が正しく付けられているかをチェックし，行番号がなければこのエラーメッセージを出します．

Disk full　[68]

意味　フロッピーディスク上に書き込むスペースがないのに書き込もうとした．

原因　• フロッピーディスク上の空き領域よりも，大きなプログラムをセーブしようとした．

• ファイルの書き込み時に，フロッピーディスクがいっぱいになった．

解説　このエラーが出たときには，そのフロッピーディスクにはそれ以上のデータは書き込めませんので，ファイルをオープンしているときはCLOSE文を実行したあとに他のフロッピーディスクに取り替えてください．

Disk I/O error　[64]

意味　フロッピーディスクとの入出力中にエラーが発生した．

原因　• フロッピーディスクがフォーマットされていない．
• フロッピーディスクに，傷やよごれがある．
• フロッピーディスクユニットに故障がある．
• ライトプロテクトされているフロッピーディスクに書き込みを行った．

解説　このエラーが，ある特定のフロッピーディスクに関してのみ出る場合には，フロッピーディスクに傷やよごれが付いていることが考えられます．また，フロッピーディスクによらずこのエラーが出る場合には，フロッピーディスクユニットが故障している可能性があります．

注意事項　コピープロテクトがかけられたソフトをコピーしようとするとこのエラーが出ることがありますが，これは故障ではありません．

Disk offline　[62]

意味　フロッピーディスクに対して入出力ができない．

原因　• フロッピーディスクが入っていないドライブにロードやセーブを行った．
• フロッピーディスクドライブのレバーが閉まっていない．

解説　このエラーは，8インチフロッピーディスクユニットを使用しているときのみ出ます．

Division by zero　[11]

意味　0による除算が実行されて，その結果がオーバーフローした．

原因　• 未定義の変数(初期値として0になっている)で除算を行った．
• 演算の結果，除数となる変数が0になっている．
• TAN関数の引数が$\pi/2$になっている．
• 0に対して負のべき乗を行った．

解説　エラーが出たときに，除数に使っている変数をPRINTして値を確かめます．0であればなぜ0になるか，プログラムの中でその変数を使って演算を行っている部分を検討してみます．除算には実数除算"/"，整数除算"¥"，剰余"MOD"の3つがあります．

Duplicate Definition　[10]

意味　同じ名前の配列を2度宣言した．

原因　一度宣言した配列を再定義した．また，宣言しないで使った配列も，添字の上限を10として自動的に宣言したことになるので，再定義はできません．

解説　一度宣言した配列は，NEW，RUN，CLEARなどの命令が実行されるまでは再定義できません．これらの命令が実行されると，他の変数もすべてクリアされてしまいます．特に配列の添字の上限だけを変えることはできないので，こういう場合は，別の名前の配列を新しく宣言して定義す

るしかありません．特定の配列だけ再定義したいときは，ERASE文を使用してその配列を消去してからDIM文を実行してください．

Duplicate label [31]

意味　同じラベル名が2つ以上存在する．

原因　異なるサブルーチンに同じラベル名を使用している．

解説　ラベルを二重定義することはできません．スクリーンエディタを使用して，プログラムを修正してください．また，このエラーでは行番号は表示されません．

Feature not available [33]

意味　利用不可能な機能を指定した．

原因
- GP-IB関係の命令を，GP-IBボードを使用せずに使おうとした．
- 拡張命令を追加せずに，CMDステートメントを実行しようとした．
- Disk versionでなければ使えない機能を使おうとした．

解説　GP-IB関係の命令などは，オプションのGP-IBボードを接続していなければ使用することはできません．

FIELD overflow [50]

意味　FIELD文において256バイト以上の大きさの領域を指定した．

原因　FIELD文において1つの文字変数のフィールド幅，もしくは1つのバッファのフィールド幅の合計値が256文字を超えている．

解説　1つのバッファの合計の文字数は256までです．FIELD文を使用する場合は，1つのバッファに対するフィールド幅の合計が256文字を超えないように設定してください．

File already exists [65]

意味　NAME文によって変更しようとした新ファイル名がすでに存在している．

原因　NAME文によって指定した新ファイル名がフロッピーディスクに存在している．

解説　NAME文によって指定した新ファイル名がフロッピーディスクに存在すると，このエラーが発生します．他のファイル名を使用するか，フロッピーディスクに存在するファイルをKILLコマンドで削除するなどしてから実行してください．

File already open [54]

意味　同じファイル番号で2度OPEN文を実行しようとした．または，すでにオープンされているファイルに対してOPEN, KILLなどを実行しようとした．

原因
- OPEN文の実行後CLOSEするのを忘れている．
- KILLの前にCLOSEするのを忘れている．

解説　オープン中のファイル番号を，他のファイルをオープンするために使うことはできません．また，オープンされているファイルをCLOSEせずに再びオープンすることもできません．いずれの場合も，ファイルをOPENしたらCLOSEするのを忘れないようにします．KILLの場合も同様です．

File not found　[53]

意味　指定したファイルが見つからない．

原因　LOAD, SAVE, KILL, OPENなどで指定したファイル名が指定したフロッピーディスク上にない．

解説　このようなエラーをなるべく避けるために，ファイル名はわかりやすい名前を付け，拡張子を付けておくとよいでしょう．

File not OPEN　[60]

意味　オープンされていないファイルを参照しようとした．

原因　まだOPEN文でファイルを割り当てていないファイル番号を使ってPRINT#などを実行した．

解説　PRINT#, INPUT#, PRINT# USING, LINE INPUT#, WRITE#文およびINPUT$, GET, PUTで使うファイル番号は，事前にOPEN文によって対応するファイルを割り当てておかなければなりません．

File write protected　[61]

意味　書き込み禁止属性が付けられているファイルに書き込もうとした．

原因
- 書き込もうとしたフロッピーディスクまたはファイルに書き込み禁止属性が付けられている．
- ミニフロッピーディスクにライトプロテクトシールが貼ってある．
- 8インチフロッピーディスクにライトプロテクトノッチを切り込んでいる．

解説　書き込もうとしたフロッピーディスク，もしくはファイルにSET文で書き込み禁止属性が付けられている場合は，大文字のPまたはR以外の属性文字を指定し，書き込み禁止属性を解除してください．SAVEコマンドのPオプションによって書き込み禁止属性が付けられている場合，それを解除する方法は用意されていません．

また，ミニフロッピーディスクにライトプロテクトシールが貼ってあったり，8インチフロッピーディスクにライトプロテクトノッチを切り込んでいると，このエラーが発生しますので，ミニフロッピーディスクの場合はシールをはがし，8インチフロッピーディスクの場合はライトプロテクトノッチに銀色のシールを貼ってください．

FOR without NEXT　[26]

意味　FOR～NEXTが正しく対応していない(FORが多い)．

原因　プログラム中のFOR～NEXT文が1対1に対応しておらず，FOR文が多い．

解説　FOR文とNEXT文とは必ず1対1に対応していなければなりません．またプログラムの見かけ上，1対1になっていても，IF文中にNEXT文がある場合に，1対1とならないことがあります．

Illegal direct　[12]

意味　ダイレクトモードで使用できない文を実行しようとした．

原因　コマンドとして使えないステートメントをダイレクトモードで使った．

解説　このエラーが表示されたステートメントはプログラムモードでしか使用できません(たとえば，DEF FNをダイレクトモードで使用した場合)．

Illegal function call　[5]

意味　ステートメントや関数において機能の呼び方が間違っている．

原因　• パラメータの値がおかしい．

• PRINT USINGでの桁指定が24桁を超えている．

• RENUMで行の順番を入れ替えようとした．

解説　原因が多岐にわたり，頻発するエラーの1つです．各関数やステートメントの使い方を区別して覚えることが大切です．プログラム中のエラーが生じるのは，ほとんどの場合，パラメータに使用している変数が演算の結果予定外の値になってしまうことによります．これはエラーが出た直後にその変数の値をPRINTしてみればすぐわかります．パラメータに使う変数名は極力わかりやすいものにして，他と区別するようにするとよいでしょう．

Input past end　[55]

意味　ファイル中のすべてのデータを読み尽くした後にINPUT#, GET等が実行された．

原因　INPUT#, INPUT$, LINE INPUT#, GETを実行する回数がファイルのデータ数より多い．

解説　INPUT#, GETなどでデータを次々と読み出すとき，ファイル中のデータ数以上読ませないようにします．ファイル中のデータ数がわからないときは，EOFあるいはLOF関数を使って，データの最後を確かめてからINPUT#, GETを実行します．

Internal error　[51]

意味　BASIC内部にエラーが生じた．

原因　BASICプログラムに，他のエラーメッセージでは表せないエラーが発生した．

解説　ハードウェアの故障(たとえばシンセサイザICの故障等)があるときにこのエラーが発生する場合があります．

Line buffer overflow　[23]

意味　RS-232C使用時に受信用バッファがオーバーフローした．

原因　• RS-232Cの受信用バッファの大きさ(256バイト)以上の文字数が送られた．

解説　RS-232Cで受信できる文字数は256までです．相手側から送られて来るデータが256文字を超え

ないようにしてください．もしこのエラーが出る場合は，転送速度を落とすとうまくいく場合があります．

Missing operand [22]

意味　ステートメント中必要なパラメータが指定されていない．

原因　• コマンドや関数で省略できないパラメータや引数が欠けている．
• 代入文の右辺がない．
• 演算子があるのに被演算子がない．

解説　コマンド・ステートメント，関数の使い方や第1章の式と演算などをもう一度確認してください．

NEXT without FOR [1]

意味　FOR～NEXTが正しく対応していない(NEXTが多い)．

原因　• FORの数がNEXTの数と合っていない．
• FOR～NEXTループの中にGOTOやGOSUBで飛び込んでくる．
• FOR～NEXTの多重ループが正しい入れ子構造になっていない．

解説　FORの数とNEXTの数は必ず一致するようにプログラミングをします．したがって，IF文中にNEXT文を置くことができない場合があります．入れ子構造を理解するためには，NEXTの後に制御変数(カウンタ)を付けておき，複数をまとめて指定しない方が見やすくなります．

No RESUME [19]

意味　RESUME文がない．

原因　エラー処理ルーチンの中にRESUME文がない．

解説　エラー処理ルーチンは必ずRESUME, END, ON ERROR GOTO 0のどれかで終わっていなければなりません．

Out of DATA [4]

意味　読むべきデータがないのにREAD文が実行された．

原因　• DATA文のデータ数が足りない．
• RESTORE文の使い方が間違っている．

解説　同じデータを何度も読ませる場合や，データをグループごとに違った行にまとめておいて扱うときはRESTORE文を使いますが，RESTORE文で指定した行番号が間違っていると，このようなエラーが発生します．READ文とDATA文の対応を正しく行ってください．

Out of memory [7]

意味　メモリ容量が足りなくなった．

原因　• プログラムが長すぎてメモリの中に収まりきらない．
• プログラムは収まっても，それを走らせるのに必要なメモリ(変数，スタック用など)が足りない．

- 配列が大きすぎて，メモリ上にその領域をとることができない．
- FOR文やGOSUB文によるネスティング(入れ子)が深くなりすぎて，スタックがいっぱいになった．
- 機械語用のメモリ領域を(すでにプログラムやファイルで使っている領域を壊すほど)大きくとろうとした．
- PRINT文で複雑な図形をペイントしようとした．

解説　このエラーが表示されたら，FRE関数を使ってフリーメモリを調べてください．プログラムが短いのにフリーメモリが異常に少ないときは，起動時に多くのバッファをとっていたり，機械語領域を指定したまま忘れている可能性があります．また変数や配列はできるだけ整数型にします．

また，間違って10 GOSUB 10のように無限にサブルーチンコールを繰り返してしまうと，戻り番地を覚えるためのスタックが無限にふくれ上がり，メモリが足りなくなってしまいます．

Out of string space　[14]

意味　文字列を格納するメモリ領域がない．

原因　文字列を扱っているときにメモリ領域がなくなった．

解説　BASICはストリングのためのスペースを自動的に確保します．したがって，ストリング領域は一定ではなく，そのときの文字列の量によって変化します．このエラーが表示されたら，文字列を少なくするか，またはフリーメモリを多くするためにプログラムを小さくするか，あるいはプログラムを分割する必要があります．

Overflow　[6]

意味　入力された数値，代入される数値，演算結果などが許される範囲を超えている．

原因
- 整数演算の結果や整数型変数に代入する値が－32768から32767の範囲外である．
- 実数演算などの結果が－1.70141E＋38から1.70141E＋38の範囲にない．
- PEEK, POKE, OUT, DIMなどで指定するパラメータや添字の値が正しい範囲にない．

解説　変数の型とその範囲，コマンド・ステートメントのパラメータの範囲，関数で扱える範囲，論理演算の範囲などを把握しておくことが必要です．

Rename across disks　[73]

意味　NAME文において，異なるフロッピーディスクドライブ間でファイルの名前を変えようとした．

原因　NAME文で指定するファイルディスクリプタのドライブ番号が〈旧ファイル名〉と〈新ファイル名〉で異なっている．

解説　ファイルディスクリプタのドライブ番号は〈旧ファイル名〉と〈新ファイル名〉の両方に指定しなければなりません．

ドライブ番号を省略した場合，ドライブ１が指定されたとみなされます．

RESUME without error [20]

意味　エラー処理ルーチンに制御を移さなかったのにRESUME文がある．

原因　• GOTOやGOSUBでエラー処理ルーチンの中へ分岐している．

• メインルーチン終了後にEND文がなく，後に続いているエラー処理ルーチンの実行が始まってしまう．

解説　メインルーチンの終わりにはEND文を置き，エラー処理ルーチンに不要意に飛び込むのを避けます．

RETURN without GOSUB [3]

意味　GOSUB～RETURNが正しく対応していない(RETURNが多い)．

原因　• サブルーチンへGOTO文で飛び込んでいる．

• メインルーチン終了後にEND文がなく，後に続いているサブルーチンの実行が始まってしまう．

解説　メインルーチンの終わりにはEND文を置くようにします．また，サブルーチンとサブルーチンの間には必ずRETURN文を置くようにしてください．

Sequential after PUT [58]

意味　PUT#の後でシーケンシャルアウトプットを行った．

原因　ランダムアクセスファイルとしてPUT#を実行してファイルにデータを書き込んだ後，PRINT#を実行した．

解説　ファイルにデータを書き込む場合，PRINT#とPUT#を混在して使用できません．シーケンシャルファイルとランダムアクセスファイルは区別して使用してください．

Sequential I/O only [59]

意味　シーケンシャルファイルの入出力以外は行ってはならない．

原因　MERGEコマンド，CHAIN文などアスキー形式のファイルしか指定できない命令においてバイナリ形式のファイルを指定した．

解説　MERGEコマンド，CHAIN MERGE文はメモリ上のプログラムと〈ファイルディスクリプタ〉によって指定されたプログラムファイルを1つにしてメモリ上に置きますが，指定するファイルは必ずアスキーセーブされていなければなりません．

String formula too complex [16]

意味　文字式が複雑すぎる．

原因　カッコを幾重にも重ねすぎるなど，文字式があまりにも複雑で解析できない．

解説　めったに起こるエラーではありませんが，複雑すぎる文字式は2つ以上に分割してください．

String too long [15]

意味　文字列が長すぎる．

原因　文字演算の結果，文字列の長さが255を超えてしまった．

解説　文字列の長さは255までと決まっているので，A$＝A$＋"LONG"などの演算を繰り返していると，A$の長さが256になった時点でエラーが発生します．文字列演算が無限ループの中にあったり，演算の要素になる文字列(特に文字型変数)が予定外の長さになっているということのないように気をつけます．どうしても256以上の長さの文字列を扱いたければ，2つ以上の変数名を使って分割処理しなければなりません．

Subscript out of range　[9]

意味　配列の添字がDIM文によって宣言されたときの大きさの範囲を超えている．

原因　• 添字に変数を使っている場合，この変数の値が演算の結果大きくなりすぎている．
- 配列の下限を1に指定しているにもかかわらず，配列の添字として0が用いられている．
- 配列の次元数を誤っている．

解説　エラーが起こったらすぐに，その添字として使っている変数の内容をPRINTして調べます．予定外に大きければ，その変数を使った演算が原因です．また，OPTION BASEを1に設定してあるときに，配列の添字に0が代入された場合にもエラーが起こります．宣言していない配列は常にその添字の上限を10とします．

Syntax error　[2]

意味　文の記述が文法どおりになっていない．

原因　• タイプミスでBASIC文法に合わないものが，プログラムにまぎれ込んでいる．
- 関数や数式が代入文の左辺にあったり，ステートメントのように単独で使われている．
- 変数名が英字で始まっていない，FNで始まっているなど．
- マルチステートメントの区切り記号":"が抜けている．
- 行番号が1から65529の範囲にない．
- 行番号の指定に変数を使おうとしている．
- IF文中で，対応するTHENのないELSEが使われている．
- コマンドのパラメータや関数の引数の個数に過不足がある．
- スクリーンエディット中に2つの行がつながってしまっている．

解説　エラーが出たときにLIST．を実行すると，ほとんどの場合，エラーのある行を示しますから，この中で原因となる間違いを探します．単純なタイプミスなどはすぐわかりますが，次のようなものはなかなか気がつかないことがあります．
- 1とI，0とO，ピリオド"."とコンマ","，コロン":"とセミコロン";"などの違い．
- 複雑な数式でカッコが正しく対応していない．
- 2つの行が接合している．
- エラーメッセージと一緒に行番号が表示されない場合，ラベルに予約語を使用している．

Tape read ERROR [27]

意味　テープからの読み込み時にエラーが発生した．

原因　テープレコーダの不良，テープの不良，録音時の不良，ロードレベルの不良等の原因で，読み込んでいるデータのフォーマットに誤りがあった．

解説　このエラーが発生した場合には，次の点を再確認してください．

- テープレコーダの音量調節は適当か(テープレコーダの音量，音質ツマミを調節してみる)．
- メモリスイッチの設定は適当か(RS-232C設定画面でDELコードを有効にする)．
- テープレコーダとの相性はよいか(他のテープレコーダを使用して読んでみる)．
- テープへの録音状態が悪くないか(他のCPU本体を使用して読み込んでみる)．

Type mismatch [13]

意味　式の左辺・右辺，関数の引数などにおいて変数の型が一致していない．

原因
- 文字型変数に数値を代入しようとした．
- 数値変数に文字列を代入しようとした．
- FOR文の制御変数に倍精度実数型を使おうとした．

解説　CHR$とASC, STR$とVALのような関数の使い方を確認してください．変数や定数について正しく理解しておくと間違いを見つけるのが容易になります．

また，FOR文の制御変数は整数型と単精度実数型しか使えません．

Undefined label [32]

意味　定義されていないラベル名を参照した．

原因　GOTO文などで分岐先を示すためにラベル名を使用しているが，該当するラベル名が存在しない．

解説　ラベル名を付ける場合には，使用できる文字の種類，文字数，ステートメント中に置かれる位置に制限がありますので，ラベル名の使用時の注意点を再確認してください(1.5　ラベル参照)．

Undefined line number [8]

意味　飛び先として必要とされる行番号が存在しない．

原因
- RENUM実行時に，参照すべき行番号がない．
- GOTO, GOSUBなどの分岐先の行番号が存在しない．
- RESTORE, RUNで指定した行番号が存在しない．

解説　GOTO文の分岐先の行をプログラム編集時に消去したまま忘れていることがあります．編集で一行消してしまうときは，そこに分岐する命令がプログラム中にないことを確かめてください．

Undefined user function [18]

意味　DEF FN文によって定義されていないユーザ定義関数を引用しようとした．

原因　DEF FN文で関数を定義せずにユーザ定義関数を呼んだ．

解説　DEF FN文で定義された関数を呼ぶためには，その前にDEF FN文で関数を定義する必要があります．定義の方法は2章のDEF FNを参照してください．

Unprintable error　[21]

意味　メッセージの定義されていないエラーを出そうとした．

原因　ERROR文により定義されていないエラーコード番号のエラーを発生した．

解説　"Unprintable error"は，BASICの拡張やユーザ定義のために残されたコード番号に割り当ててあるもので，21の他，34～49および74～255のコード番号に対応しています．

ERROR文でユーザが定義したエラーメッセージを出すには，エラー処理ルーチンでその処理を行わなければなりません．

WEND without WHILE　[30]

意味　WHILE～WENDが正しく対応していない(WENDが多い)．

原因　対応するWHILE文がないにもかかわらずWEND文がプログラム中に書かれている．

解説　WEND文はWHILE文と対になり，ループの終わりを示す働きをし，必ず1対1に対応していなければなりません．

WHILE without WEND　[29]

意味　WHILE～WENDが正しく対応していない(WHILEが多い)．

原因　WHILE文を指定したが，対応するWEND文がプログラム中に書かれていない．

解説　WEND文はWHILE文と対になりループの終わりを示す働きをしますので，省略することはできません．

資料7　半角文字/全角文字コード変換表

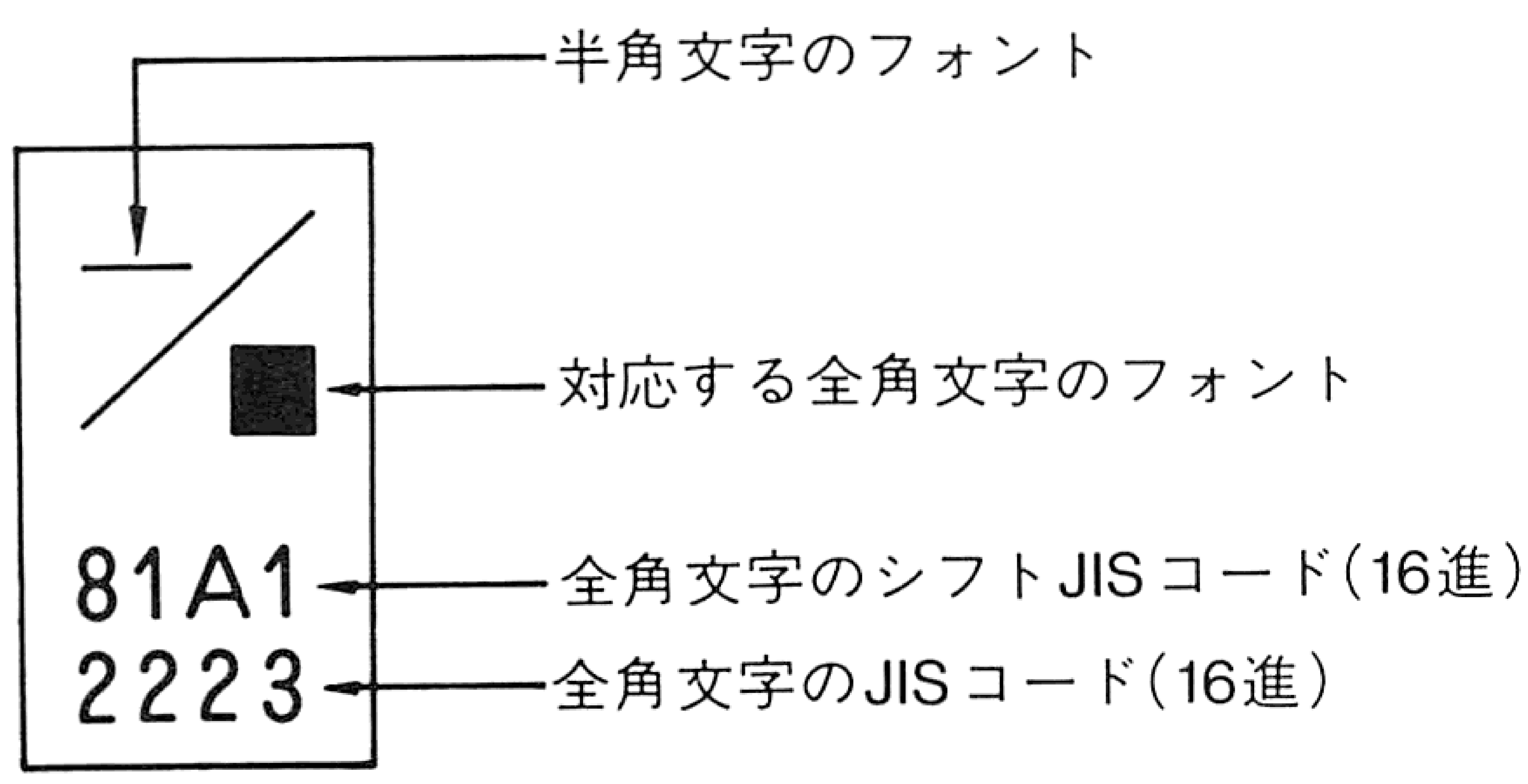
AKCNV$関数，KACNV$関数は，この変換表に従って半角文字と全角文字の変換を行います．

表のみかた

• 半角文字のコードは，最上行の0～Fを上位4ビット最左列の0～Fを下位4ビットとする1バイトで表されます．たとえば半角文字Aのコードは41Hとなります．

• 半角文字はAKCNV$関数によって文字の下に示される4桁のシフトJISコード，およびJISコードを持つ全角文字に変換されます．たとえば，半角文字のAはシフトJISコード　8260H，JISコード　2341Hの全角文字のAに変換されます．対応する全角文字のフォントが異なるときはスラッシュの左上が半角文字のフォント，右下が全角文字のフォントを示します．

例）



• AKCNV$関数で半角文字を全角文字に変換する場合は，アミのかけてある部分とかけてない部分(つまり，黒く塗りつぶしてない部分)が変換の対象になります．

• 全角文字を，KACNV$関数によって半角文字に変換する場合は，アミのかけてある部分だけが変換の対象になります．

• 黒く塗りつぶしてある部分は，全角→半角，半角→全角どちらの変換の対象にもなりません．

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	／■ 81A1 2223	DE ／■ 81A1 2223	8140 2121	0 824F 2330	@ 8197 2177	P 826F 2350	／■ 81A1 2223	p 8290 2370	／■ 81A1 2223		／■ 81A1 2223	ー 815B 213C	タ 835E 253F	ミ 837E 255F		
1	SH ／■ 81A1 2223	D1 ／■ 81A1 2223	! 8149 212A	1 8250 2331	A 8260 2341	Q 8270 2351	a 8281 2361	q 8291 2371			。 8142 2123	ア 8341 2522	チ 8360 2541	ム 8380 2560		
2	SX ／■ 81A1 2223	D2 ／■ 81A1 2223	” 8168 2149	2 8251 2332	B 8261 2342	R 8271 2352	b 8282 2362	r 8292 2372			「 8175 2156	イ 8343 2524	ツ 8363 2544	メ 8381 2561		
3	EX ／■ 81A1 2223	D3 ／■ 81A1 2223	# 8194 2174	3 8252 2333	C 8262 2343	S 8272 2353	c 8283 2363	s 8293 2373			」 8176 2157	ウ 8345 2526	テ 8365 2546	モ 8382 2562		
4	ET ／■ 81A1 2223	D4 ／■ 81A1 2223	$ 8190 2170	4 8253 2334	D 8263 2344	T 8273 2354	d 8284 2364	t 8294 2374			、 8141 2122	エ 8347 2528	ト 8367 2548	ヤ 8384 2564		
5	EQ ／■ 81A1 2223	NK ／■ 81A1 2223	% 8193 2173	5 8254 2335	E 8264 2345	U 8274 2355	e 8285 2365	u 8295 2375			・ 8145 2126	オ 8349 252A	ナ 8369 254A	ユ 8386 2566		
6	AK ／■ 81A1 2223	SN ／■ 81A1 2223	& 8195 2175	6 8255 2336	F 8265 2346	V 8275 2356	f 8286 2366	v 8296 2376			ヲ 8392 2572	カ 834A 252B	ニ 836A 254B	ヨ 8388 2568		
7	BL ／■ 81A1 2223	EB ／■ 81A1 2223	’ 8166 2147	7 8256 2337	G 8266 2347	W 8276 2357	g 8287 2367	w 8297 2377			ァ 8340 2521	キ 834C 252D	ヌ 836B 254C	ラ 8389 2569		
8	BS ／■ 81A1 2223	CN ／■ 81A1 2223	(8169 214A	8 8257 2338	H 8267 2348	X 8277 2358	h 8288 2368	x 8298 2378			ィ 8342 2523	ク 834E 252F	ネ 836C 254D	リ 838A 256A		
9	HT ／■ 81A1 2223	EM ／■ 81A1 2223	) 816A 214B	9 8258 2339	I 8268 2349	Y 8278 2359	i 8289 2369	y 8299 2379			ゥ 8344 2525	ケ 8350 2531	ノ 836D 254E	ル 838B 256B		
A		SB ／■ 81A1 2223	* 8196 2176	: 8146 2127	J 8269 234A	Z 8279 235A	j 828A 236A	z 829A 237A			ェ 8346 2527	コ 8352 2533	ハ 836E 254F	レ 838C 256C		
B	HM ／■ 81A1 2223		+ 817B 215C	; 8147 2128	K 826A 234B	[816D 214E	k 828B 236B	{ 816F 2150			ォ 8348 2529	サ 8354 2535	ヒ 8371 2552	ロ 838D 256D		
C	CL ／■ 81A1 2223	→ 81A8 222A	, 8143 2124	< 8183 2163	L 826B 234C	¥ 818F 216F	l 828C 236C	¦ 8162 2143			ャ 8383 2563	シ 8356 2537	フ 8374 2555	ワ 838F 256F		
D		← 81A9 222B	― 817C 215D	= 8181 2161	M 826C 234D	] 816E 214F	m 828D 236D	} 8170 2151			ュ 8385 2565	ス 8358 2539	ヘ 8377 2558	ン 8393 2573		／■ 81A1 2223
E	SO ／■ 81A1 2223	↑ 81AA 222C	. 8144 2125	> 8184 2164	N 826D 234E	^ 814F 2130	n 828E 236E	~ 8160 2141			ョ 8387 2567	セ 835A 253B	ホ 837A 255B	゛ 814A 212B		／■ 81A1 2223
F	SI ／■ 81A1 2223	↓ 81AB 222D	／ 815E 213F	? 8148 2129	O 826E 234F	_ 8151 2132	o 828F 236F	／■ 81A1 2223			ッ 8362 2543	ソ 835C 253D	マ 837D 255E	゜ 814B 212C		／■ 81A1 2223

資料8　日本語コード

JIS第1水準(半角文字，1/4角文字)

※　コードは16進法で表現されています．

たとえば，半角文字"&"のコードは0020H＋6H＝0026Hとなります．

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
半角文字	0020		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
	0030	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
	0040	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	0050	P	Q	R	S	T	U	V	W	X	Y	Z	[	¥	]	^	—
	0060		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
	0070	p	q	r	s	t	u	v	w	x	y	z	{	¦	}	~	
	0080		。	「	」	、	・	を	ぁ	ぃ	ぅ	ぇ	ぉ	ゃ	ゅ	ょ	っ
	0090	ー	あ	い	う	え	お	か	き	く	け	こ	さ	し	す	せ	そ
	00A0		。	「	」		・	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ
	00B0	ー	ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ	サ	シ	ス	セ	ソ
	00C0	タ	チ	ツ	テ	ト	ナ	ニ	ヌ	ネ	ノ	ハ	ヒ	フ	ヘ	ホ	マ
	00D0	ミ	ム	メ	モ	ヤ	ユ	ヨ	ラ	リ	ル	レ	ロ	ワ	ン	゛	゜
	00E0	た	ち	つ	て	と	な	に	ぬ	ね	の	は	ひ	ふ	へ	ほ	ま
	00F0	み	む	め	も	や	ゆ	よ	ら	り	る	れ	ろ	わ	ん	゛	゜
1/4角文字	0100		S_H	S_X	E_X	E_T	E_Q	A_K	B_L	B_S	H_T	L_F	H_M	C_L	C_R	S_O	S_I
	0110	D_E	D_1	D_2	D_3	D_4	N_K	S_N	E_B	C_N	E_M	S_B	E_C	→	←	↑	↓
	0120		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
	0130	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
	0140	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	0150	P	Q	R	S	T	U	V	W	X	Y	Z	[	¥	]	^	—
	0160		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
	0170	p	q	r	s	t	u	v	w	x	y	z	{	¦	}	~	
	0180	▁	▂	▃	▄	▅	▆	▇	█	▏	▎	▍	▌	▋	▊	▉	┼
	0190	┴	┬	┤	├	▔	─	│	▕	┌	┐	└	┘	╭	╮	╰	╯
	01A0		。	「	」	、	・	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ
	01B0	ー	ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ	サ	シ	ス	セ	ソ
	01C0	タ	チ	ツ	テ	ト	ナ	ニ	ヌ	ネ	ノ	ハ	ヒ	フ	ヘ	ホ	マ
	01D0	ミ	ム	メ	モ	ヤ	ユ	ヨ	ラ	リ	ル	レ	ロ	ワ	ン	゛	゜
	01E0	═	╞	╪	╡	◢	◣	◥	◤	♠	♥	♦	♣	●	○	╱	╲
	01F0	×	円	年	月	日	時	分	秒								
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

注意：半角文字および1/4角文字を出力するには，PUT@ KANJI文を使用します．

JIS第１水準（全角文字）

※　コードは16進法で表現されています．

　たとえば，"愛"のシフトJIS(SJIS)コードは88A0H＋4H＝88A4H, JISコードは3022H＋4H＝3026H となります．

	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
記号	2120	813F			、	。	，	．	・	：	；	？	！	゛	゜	´	｀	¨
	2130	814F	＾	￣	＿	ヽ	ヾ	ゝ	ゞ	〃	仝	々	〆	〇	ー	―	‐	／
	2140	815F	＼	～	‖	｜	…	‥	‘	’	“	”	（	）	〔	〕	［	］
	2150	816F	｛	｝	〈	〉	《	》	「	」	『	』	【	】	＋	－	±	×
	2160	8180	÷	＝	≠	＜	＞	≦	≧	∞	∴	♂	♀	°	′	″	℃	￥
	2170	8190	＄	￠	￡	％	＃	＆	＊	＠	§	☆	★	○	●	◎	◇	◆
	2220	819E			□	■	△	▲	▽	▼	※	〒	→	←	↑	↓	〓	
英・数字	2330	824F	0	1	2	3	4	5	6	7	8	9						
	2340	825F		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	2350	826F	P	Q	R	S	T	U	V	W	X	Y	Z					
	2360	8280		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
	2370	8290	p	q	r	s	t	u	v	w	x	y	z					
ひらがな	2420	829E		ぁ	あ	ぃ	い	ぅ	う	ぇ	え	ぉ	お	か	が	き	ぎ	く
	2430	82AE	ぐ	け	げ	こ	ご	さ	ざ	し	じ	す	ず	せ	ぜ	そ	ぞ	た
	2440	82BE	だ	ち	ぢ	っ	つ	づ	て	で	と	ど	な	に	ぬ	ね	の	は
	2450	82CE	ば	ぱ	ひ	び	ぴ	ふ	ぶ	ぷ	へ	べ	ぺ	ほ	ぼ	ぽ	ま	み
	2460	82DE	む	め	も	ゃ	や	ゅ	ゆ	ょ	よ	ら	り	る	れ	ろ	ゎ	わ
	2470	82EE	ゐ	ゑ	を	ん												
カタカナ	2520	833F		ァ	ア	ィ	イ	ゥ	ウ	ェ	エ	ォ	オ	カ	ガ	キ	ギ	ク
	2530	834F	グ	ケ	ゲ	コ	ゴ	サ	ザ	シ	ジ	ス	ズ	セ	ゼ	ソ	ゾ	タ
	2540	835F	ダ	チ	ヂ	ッ	ツ	ヅ	テ	デ	ト	ド	ナ	ニ	ヌ	ネ	ノ	ハ
	2550	836F	バ	パ	ヒ	ビ	ピ	フ	ブ	プ	ヘ	ベ	ペ	ホ	ボ	ポ	マ	ミ
	2560	8380	ム	メ	モ	ャ	ヤ	ュ	ユ	ョ	ヨ	ラ	リ	ル	レ	ロ	ヮ	ワ
	2570	8390	ヰ	ヱ	ヲ	ン	ヴ	ヵ	ヶ									
	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

注意：空白文字のコードは，2121H(JISコード)です．

	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ギリシア文字	2620	839E		Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο
	2630	83AE	Π	Ρ	Σ	Τ	Υ	Φ	Χ	Ψ	Ω							
	2640	83BE		α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο
	2650	83CE	π	ρ	σ	τ	υ	φ	χ	ψ	ω							
ロシア文字	2720	843F		А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н
	2730	844F	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э
	2740	845F	Ю	Я														
	2750	846F		а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н
	2760	8480	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э
	2770	8490	ю	я														
	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	JIS コード	シフトJIS コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ア	3020	889E		亜	唖	娃	阿	哀	愛	挨	姶	逢	葵	茜	穐	悪	握	渥
	3030	88AE	旭	葦	芦	鯵	梓	圧	斡	扱	宛	姐	虻	飴	絢	綾	鮎	或
	3040	88BE	粟	袷	安	庵	按	暗	案	闇	鞍	杏						
イ	3040	88BE											以	伊	位	依	偉	囲
	3050	88CE	夷	委	威	尉	惟	意	慰	易	椅	為	畏	異	移	維	緯	胃
	3060	88DE	萎	衣	謂	違	遺	医	井	亥	域	育	郁	磯	一	壱	溢	逸
	3070	88EE	稲	茨	芋	鰯	允	印	咽	員	因	姻	引	飲	淫	胤	蔭	
	3120	893F		院	陰	隠	韻	吋										
ウ	3120	893F							右	宇	烏	羽	迂	雨	卯	鵜	窺	丑
	3130	894F	碓	臼	渦	嘘	唄	欝	蔚	鰻	姥	厩	浦	瓜	閏	噂	云	運
	3140	895F	雲															
エ	3140	895F		荏	餌	叡	営	嬰	影	映	曳	栄	永	泳	洩	瑛	盈	穎
	3150	896F	頴	英	衛	詠	鋭	液	疫	益	駅	悦	謁	越	閲	榎	厭	円
	3160	8980	園	堰	奄	宴	延	怨	掩	援	沿	演	炎	焔	煙	燕	猿	縁
	3170	8990	艶	苑	薗	遠	鉛	鴛	塩									
オ	3170	8990								於	汚	甥	凹	央	奥	往	応	
	3220	899E		押	旺	横	欧	殴	王	翁	襖	鴬	鴎	黄	岡	沖	荻	億
	3230	89AE	屋	憶	臆	桶	牡	乙	俺	卸	恩	温	穏	音				
カ	3230	89AE													下	化	仮	何
	3240	89BE	伽	価	佳	加	可	嘉	夏	嫁	家	寡	科	暇	果	架	歌	河
	3250	89CE	火	珂	禍	禾	稼	箇	花	苛	茄	荷	華	菓	蝦	課	嘩	貨
	3260	89DE	迦	過	霞	蚊	俄	峨	我	牙	画	臥	芽	蛾	賀	雅	餓	駕
	3270	89EE	介	会	解	回	塊	壊	廻	快	怪	悔	恢	懐	戒	拐	改	
	3320	8A3F		魁	晦	械	海	灰	界	皆	絵	芥	蟹	開	階	貝	凱	劾
	3330	8A4F	外	咳	害	崖	慨	概	涯	碍	蓋	街	該	鎧	骸	浬	馨	蛙
	3340	8A5F	垣	柿	蛎	鈎	劃	嚇	各	廓	拡	撹	格	核	殻	獲	確	穫
	3350	8A6F	覚	角	赫	較	郭	閣	隔	革	学	岳	楽	額	顎	掛	笠	樫
	3360	8A80	橿	梶	鰍	潟	割	喝	恰	括	活	渇	滑	葛	褐	轄	且	鰹
	JIS コード	シフトJIS コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
カ	3370	8A90	叶	椛	樺	鞄	株	兜	竃	蒲	釜	鎌	噛	鴨	栢	茅	萱	
	3420	8A9E		粥	刈	苅	瓦	乾	侃	冠	寒	刊	勘	勧	巻	喚	堪	姦
	3430	8AAE	完	官	寛	干	幹	患	感	慣	憾	換	敢	柑	桓	棺	款	歓
	3440	8ABE	汗	漢	澗	潅	環	甘	監	看	竿	管	簡	緩	缶	翰	肝	艦
	3450	8ACE	莞	観	諌	貫	還	鑑	間	閑	関	陥	韓	館	舘	丸	含	岸
	3460	8ADE	巌	玩	癌	眼	岩	翫	贋	雁	頑	顔	願					
キ	3460	8ADE												企	伎	危	喜	器
	3470	8AEE	基	奇	嬉	寄	岐	希	幾	忌	揮	机	旗	既	期	棋	棄	
	3520	8B3F		機	帰	毅	気	汽	畿	祈	季	稀	紀	徽	規	記	貴	起
	3530	8B4F	軌	輝	飢	騎	鬼	亀	偽	儀	妓	宜	戯	技	擬	欺	犠	疑
	3540	8B5F	祇	義	蟻	誼	議	掬	菊	鞠	吉	吃	喫	桔	橘	詰	砧	杵
	3550	8B6F	黍	却	客	脚	虐	逆	丘	久	仇	休	及	吸	宮	弓	急	救
	3560	8B80	朽	求	汲	泣	灸	球	究	窮	笈	級	糾	給	旧	牛	去	居
	3570	8B90	巨	拒	拠	挙	渠	虚	許	距	鋸	漁	禦	魚	亨	享	京	
	3620	8B9E		供	侠	僑	兇	競	共	凶	協	匡	卿	叫	喬	境	峡	強
	3630	8BAE	彊	怯	恐	恭	挟	教	橋	況	狂	狭	矯	胸	脅	興	蕎	郷
	3640	8BBE	鏡	響	饗	驚	仰	凝	尭	暁	業	局	曲	極	玉	桐	粁	僅
	3650	8BCE	勤	均	巾	錦	斤	欣	欽	琴	禁	禽	筋	緊	芹	菌	衿	襟
	3660	8BDE	謹	近	金	吟	銀											
ク	3660	8BDE						九	倶	句	区	狗	玖	矩	苦	躯	駆	駈
	3670	8BEE	駒	具	愚	虞	喰	空	偶	寓	遇	隅	串	櫛	釧	屑	屈	
	3720	8C3F		掘	窟	沓	靴	轡	窪	熊	隈	粂	栗	繰	桑	鍬	勲	君
	3730	8C4F	薫	訓	群	軍	郡											
ケ	3730	8C4F						卦	袈	祁	係	傾	刑	兄	啓	圭	珪	型
	3740	8C5F	契	形	径	恵	慶	慧	憩	掲	携	敬	景	桂	渓	畦	稽	系
	3750	8C6F	経	継	繋	罫	茎	荊	蛍	計	詣	警	軽	頚	鶏	芸	迎	鯨
	3760	8C80	劇	戟	撃	激	隙	桁	傑	欠	決	潔	穴	結	血	訣	月	件
	3770	8C90	倹	倦	健	兼	券	剣	喧	圏	堅	嫌	建	憲	懸	拳	捲	
	3820	8C9E		検	権	牽	犬	献	研	硯	絹	県	肩	見	謙	賢	軒	遣
	3830	8CAE	鍵	険	顕	験	鹸	元	原	厳	幻	弦	減	源	玄	現	絃	舷
	3840	8CBE	言	諺	限													
	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
コ	3840	8CBE				乎	個	古	呼	固	姑	孤	己	庫	弧	戸	故	枯
	3850	8CCE	湖	狐	糊	袴	股	胡	菰	虎	誇	跨	鈷	雇	顧	鼓	五	互
	3860	8CDE	伍	午	呉	吾	娯	後	御	悟	梧	檎	瑚	碁	語	誤	護	醐
	3870	8CEE	乞	鯉	交	佼	侯	候	倖	光	公	功	効	勾	厚	口	向	
	3920	8D3F		后	喉	坑	垢	好	孔	孝	宏	工	巧	巷	幸	広	庚	康
	3930	8D4F	弘	恒	慌	抗	拘	控	攻	昂	晃	更	杭	校	梗	構	江	洪
	3940	8D5F	浩	港	溝	甲	皇	硬	稿	糠	紅	紘	絞	綱	耕	考	肯	肱
	3950	8D6F	腔	膏	航	荒	行	衡	講	貢	購	郊	酵	鉱	砿	鋼	閤	降
	3960	8D80	項	香	高	鴻	剛	劫	号	合	壕	拷	濠	豪	轟	麹	克	刻
	3970	8D90	告	国	穀	酷	鵠	黒	獄	漉	腰	甑	忽	惚	骨	狛	込	
	3A20	8D9E		此	頃	今	困	坤	墾	婚	恨	懇	昏	昆	根	梱	混	痕
	3A30	8DAE	紺	艮	魂													
サ	3A30	8DAE				些	佐	叉	唆	嵯	左	差	査	沙	瑳	砂	詐	鎖
	3A40	8DBE	裟	坐	座	挫	債	催	再	最	哉	塞	妻	宰	彩	才	採	栽
	3A50	8DCE	歳	済	災	采	犀	砕	砦	祭	斎	細	菜	裁	載	際	剤	在
	3A60	8DDE	材	罪	財	冴	坂	阪	堺	榊	肴	咲	崎	埼	碕	鷺	作	削
	3A70	8DEE	咋	搾	昨	朔	柵	窄	策	索	錯	桜	鮭	笹	匙	冊	刷	
	3B20	8E3F		察	拶	撮	擦	札	殺	薩	雑	皐	鯖	捌	錆	鮫	皿	晒
	3B30	8E4F	三	傘	参	山	惨	撒	散	桟	燦	珊	産	算	纂	蚕	讃	賛
	3B40	8E5F	酸	餐	斬	暫	残											
シ	3B40	8E5F						仕	仔	伺	使	刺	司	史	嗣	四	士	始
	3B50	8E6F	姉	姿	子	屍	市	師	志	思	指	支	孜	斯	施	旨	枝	止
	3B60	8E80	死	氏	獅	祉	私	糸	紙	紫	肢	脂	至	視	詞	詩	試	誌
	3B70	8E90	諮	資	賜	雌	飼	歯	事	似	侍	児	字	寺	慈	持	時	
	3C20	8E9E		次	滋	治	爾	璽	痔	磁	示	而	耳	自	蒔	辞	汐	鹿
	3C30	8EAE	式	識	鴫	竺	軸	宍	雫	七	叱	執	失	嫉	室	悉	湿	漆
	3C40	8EBE	疾	質	実	蔀	篠	偲	柴	芝	屡	蕊	縞	舎	写	射	捨	赦
	3C50	8ECE	斜	煮	社	紗	者	謝	車	遮	蛇	邪	借	勺	尺	杓	灼	爵
	3C60	8EDE	酌	釈	錫	若	寂	弱	惹	主	取	守	手	朱	殊	狩	珠	種
	3C70	8EEE	腫	趣	酒	首	儒	受	呪	寿	授	樹	綬	需	囚	収	周	
	3D20	8F3F		宗	就	州	修	愁	拾	洲	秀	秋	終	繍	習	臭	舟	蒐
	3D30	8F4F	衆	襲	讐	蹴	輯	週	酋	酬	集	醜	什	住	充	十	従	戎
	3D40	8F5F	柔	汁	渋	獣	縦	重	銃	叔	夙	宿	淑	祝	縮	粛	塾	熟
	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
シ	3D50	8F6F	出	術	述	俊	峻	春	瞬	竣	舜	駿	准	循	旬	楯	殉	淳
	3D60	8F80	準	潤	盾	純	巡	遵	醇	順	処	初	所	暑	曙	渚	庶	緒
	3D70	8F90	署	書	薯	藷	諸	助	叙	女	序	徐	恕	鋤	除	傷	償	
	3E20	8F9E		勝	匠	升	召	哨	商	唱	嘗	奨	妾	娼	宵	将	小	少
	3E30	8FAE	尚	庄	床	廠	彰	承	抄	招	掌	捷	昇	昌	昭	晶	松	梢
	3E40	8FBE	樟	樵	沼	消	渉	湘	焼	焦	照	症	省	硝	礁	祥	称	章
	3E50	8FCE	笑	粧	紹	肖	菖	蒋	蕉	衝	裳	訟	証	詔	詳	象	賞	醤
	3E60	8FDE	鉦	鍾	鐘	障	鞘	上	丈	丞	乗	冗	剰	城	場	壌	嬢	常
	3E70	8FEE	情	擾	条	杖	浄	状	畳	穣	蒸	譲	醸	錠	嘱	埴	飾	
	3F20	903F		拭	植	殖	燭	織	職	色	触	食	蝕	辱	尻	伸	信	侵
	3F30	904F	唇	娠	寝	審	心	慎	振	新	晋	森	榛	浸	深	申	疹	真
	3F40	905F	神	秦	紳	臣	芯	薪	親	診	身	辛	進	針	震	人	仁	刃
	3F50	906F	塵	壬	尋	甚	尽	腎	訊	迅	陣	靭						
ス	3F50	906F											笥	諏	須	酢	図	厨
	3F60	9080	逗	吹	垂	帥	推	水	炊	睡	粋	翠	衰	遂	酔	錐	錘	随
	3F70	9090	瑞	髄	崇	嵩	数	枢	趨	雛	据	杉	椙	菅	頗	雀	裾	
	4020	909E		澄	摺	寸												
セ	4020	909E					世	瀬	畝	是	凄	制	勢	姓	征	性	成	政
	4030	90AE	整	星	晴	棲	栖	正	清	牲	生	盛	精	聖	声	製	西	誠
	4040	90BE	誓	請	逝	醒	青	静	斉	税	脆	隻	席	惜	戚	斥	昔	析
	4050	90CE	石	積	籍	績	脊	責	赤	跡	蹟	碩	切	拙	接	摂	折	設
	4060	90DE	窃	節	説	雪	絶	舌	蝉	仙	先	千	占	宣	専	尖	川	戦
	4070	90EE	扇	撰	栓	栴	泉	浅	洗	染	潜	煎	煽	旋	穿	箭	線	
	4120	913F		繊	羨	腺	舛	船	薦	詮	賎	践	選	遷	銭	銑	閃	鮮
	4130	914F	前	善	漸	然	全	禅	繕	膳	糎							
ソ	4130	914F										噌	塑	岨	措	曾	曽	楚
	4140	915F	狙	疏	疎	礎	祖	租	粗	素	組	蘇	訴	阻	遡	鼠	僧	創
	4150	916F	双	叢	倉	喪	壮	奏	爽	宋	層	匝	惣	想	捜	掃	挿	掻
	4160	9180	操	早	曹	巣	槍	槽	漕	燥	争	痩	相	窓	糟	総	綜	聡
	4170	9190	草	荘	葬	蒼	藻	装	走	送	遭	鎗	霜	騒	像	増	憎	
	4220	919E		臓	蔵	贈	造	促	側	則	即	息	捉	束	測	足	速	俗
	4230	91AE	属	賊	族	続	卒	袖	其	揃	存	孫	尊	損	村	遜		
	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
タ	4230	91AE															他	多
	4240	91BE	太	汰	詑	唾	堕	妥	惰	打	柁	舵	楕	陀	駄	騨	体	堆
	4250	91CE	対	耐	岱	帯	待	怠	態	戴	替	泰	滞	胎	腿	苔	袋	貸
	4260	91DE	退	逮	隊	黛	鯛	代	台	大	第	醍	題	鷹	滝	瀧	卓	啄
	4270	91EE	宅	托	択	拓	沢	濯	琢	託	鐸	濁	諾	茸	凧	蛸	只	
	4320	923F		叩	但	達	辰	奪	脱	巽	竪	辿	棚	谷	狸	鱈	樽	誰
	4330	924F	丹	単	嘆	坦	担	探	旦	歎	淡	湛	炭	短	端	箪	綻	耽
	4340	925F	胆	蛋	誕	鍛	団	壇	弾	断	暖	檀	段	男	談			
チ	4340	925F														値	知	地
	4350	926F	弛	恥	智	池	痴	稚	置	致	蜘	遅	馳	築	畜	竹	筑	蓄
	4360	9280	逐	秩	窒	茶	嫡	着	中	仲	宙	忠	抽	昼	柱	注	虫	衷
	4370	9290	註	酎	鋳	駐	樗	瀦	猪	苧	著	貯	丁	兆	凋	喋	寵	帖
	4420	929E			帳	庁	弔	張	彫	徴	懲	挑	暢	朝	潮	牒	町	眺
	4430	92AE	聴	脹	腸	蝶	調	諜	超	跳	銚	長	頂	鳥	勅	捗	直	朕
	4440	92BE	沈	珍	賃	鎮	陳											
ツ	4440	92BE						津	墜	椎	槌	追	鎚	痛	通	塚	栂	掴
	4450	92CE	槻	佃	漬	柘	辻	蔦	綴	鍔	椿	潰	坪	壷	嬬	紬	爪	吊
	4460	92DE	釣	鶴														
テ	4460	92DE			亭	低	停	偵	剃	貞	呈	堤	定	帝	底	庭	廷	弟
	4470	92EE	悌	抵	挺	提	梯	汀	碇	禎	程	締	艇	訂	諦	蹄	逓	
	4520	933F		邸	鄭	釘	鼎	泥	摘	擢	敵	滴	的	笛	適	鏑	溺	哲
	4530	934F	徹	撤	轍	迭	鉄	典	填	天	展	店	添	纏	甜	貼	転	顛
	4540	935F	点	伝	殿	澱	田	電										
ト	4540	935F							兎	吐	堵	塗	妬	屠	徒	斗	杜	渡
	4550	936F	登	菟	賭	途	都	鍍	砥	砺	努	度	土	奴	怒	倒	党	冬
	4560	9380	凍	刀	唐	塔	塘	套	宕	島	嶋	悼	投	搭	東	桃	梼	棟
	4570	9390	盗	淘	湯	涛	灯	燈	当	痘	祷	等	答	筒	糖	統	到	
	4620	939E		董	蕩	藤	討	謄	豆	踏	逃	透	鐙	陶	頭	騰	闘	働
	4630	93AE	動	同	堂	導	憧	撞	洞	瞳	童	胴	萄	道	銅	峠	鴇	匿
	4640	93BE	得	徳	涜	特	督	禿	篤	毒	独	読	栃	橡	凸	突	椴	届
	4650	93CE	鳶	苫	寅	酉	瀞	噸	屯	惇	敦	沌	豚	遁	頓	呑	曇	鈍
	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	JIS コード	シフトJIS コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ナ	4660	93DE	奈	那	内	乍	凪	薙	謎	灘	捺	鍋	楢	馴	縄	畷	南	楠
	4670	93EE	軟	難	汝													
ニ	4670	93EE				二	尼	弐	迩	匂	賑	肉	虹	廿	日	乳	入	
	4720	943F		如	尿	韮	任	妊	忍	認								
ヌ	4720	943F									濡							
ネ	4720	943F										禰	祢	寧	葱	猫	熱	年
	4730	944F	念	捻	撚	燃	粘											
ノ	4730	944F						乃	廼	之	埜	嚢	悩	濃	納	能	脳	膿
	4740	945F	農	覗	蚤													
ハ	4740	945F				巴	把	播	覇	杷	波	派	琶	破	婆	罵	芭	馬
	4750	946F	俳	廃	拝	排	敗	杯	盃	牌	背	肺	輩	配	倍	培	媒	梅
	4760	9480	楳	煤	狽	買	売	賠	陪	這	蠅	秤	矧	萩	伯	剥	博	拍
	4770	9490	柏	泊	白	箔	粕	舶	薄	迫	曝	漠	爆	縛	莫	駁	麦	
	4820	949E		函	箱	硲	箸	肇	筈	櫨	幡	肌	畑	畠	八	鉢	溌	発
	4830	94AE	醗	髪	伐	罰	抜	筏	閥	鳩	噺	塙	蛤	隼	伴	判	半	反
	4840	94BE	叛	帆	搬	斑	板	氾	汎	版	犯	班	畔	繁	般	藩	販	範
	4850	94CE	釆	煩	頒	飯	挽	晩	番	盤	磐	蕃	蛮					
ヒ	4850	94CE												匪	卑	否	妃	庇
	4860	94DE	彼	悲	扉	批	披	斐	比	泌	疲	皮	碑	秘	緋	罷	肥	被
	4870	94EE	誹	費	避	非	飛	樋	簸	備	尾	微	枇	毘	琵	眉	美	
	4920	953F		鼻	柊	稗	匹	疋	髭	彦	膝	菱	肘	弼	必	畢	筆	逼
	4930	954F	桧	姫	媛	紐	百	謬	俵	彪	標	氷	漂	瓢	票	表	評	豹
	4940	955F	廟	描	病	秒	苗	錨	鋲	蒜	蛭	鰭	品	彬	斌	浜	瀕	貧
	4950	956F	賓	頻	敏	瓶												
フ	4950	956F					不	付	埠	夫	婦	富	冨	布	府	怖	扶	敷
	4960	9580	斧	普	浮	父	符	腐	膚	芙	譜	負	賦	赴	阜	附	侮	撫
	4970	9590	武	舞	葡	蕪	部	封	楓	風	葺	蕗	伏	副	復	幅	服	
	4A20	959E		福	腹	複	覆	淵	弗	払	沸	仏	物	鮒	分	吻	噴	墳
	JIS コード	シフトJIS コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
フ	4A30	95AE	憤	扮	焚	奮	粉	糞	紛	雰	文	聞						
ヘ	4A30	95AE											丙	併	兵	塀	幣	平
	4A40	95BE	弊	柄	並	蔽	閉	陛	米	頁	僻	壁	癖	碧	別	瞥	蔑	箆
	4A50	95CE	偏	変	片	篇	編	辺	返	遍	便	勉	娩	弁	鞭			
ホ	4A50	95CE														保	舗	鋪
	4A60	95DE	圃	捕	歩	甫	補	輔	穂	募	墓	慕	戊	暮	母	簿	菩	倣
	4A70	95EE	俸	包	呆	報	奉	宝	峰	峯	崩	庖	抱	捧	放	方	朋	
	4B20	963F		法	泡	烹	砲	縫	胞	芳	萌	蓬	蜂	褒	訪	豊	邦	鋒
	4B30	964F	飽	鳳	鵬	乏	亡	傍	剖	坊	妨	帽	忘	忙	房	暴	望	某
	4B40	965F	棒	冒	紡	肪	膨	謀	貌	貿	鉾	防	吠	頬	北	僕	卜	墨
	4B50	966F	撲	朴	牧	睦	穆	釦	勃	没	殆	堀	幌	奔	本	翻	凡	盆
マ	4B60	9680	摩	磨	魔	麻	埋	妹	昧	枚	毎	哩	槙	幕	膜	枕	鮪	柾
	4B70	9690	鱒	桝	亦	俣	又	抹	末	沫	迄	侭	繭	麿	万	慢	満	
	4C20	969E		漫	蔓													
ミ	4C20	969E				味	未	魅	巳	箕	岬	密	蜜	湊	蓑	稔	脈	妙
	4C30	96AE	粍	民	眠													
ム	4C30	96AE				務	夢	無	牟	矛	霧	鵡	椋	婿	娘			
メ	4C30	96AE														冥	名	命
	4C40	96BE	明	盟	迷	銘	鳴	姪	牝	滅	免	棉	綿	緬	面	麺		
モ	4C40	96BE															摸	模
	4C50	96CE	茂	妄	孟	毛	猛	盲	網	耗	蒙	儲	木	黙	目	杢	勿	餅
	4C60	96DE	尤	戻	籾	貰	問	悶	紋	門	匁							
ヤ	4C60	96DE										也	冶	夜	爺	耶	野	弥
	4C70	96EE	矢	厄	役	約	薬	訳	躍	靖	柳	薮	鑓					
ユ	4C70	96EE												愉	愈	油	癒	
	4D20	973F		諭	輸	唯	佑	優	勇	友	宥	幽	悠	憂	揖	有	柚	湧
	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ユ	4D30	974F	涌	猶	猷	由	祐	裕	誘	遊	邑	郵	雄	融	夕			
ヨ	4D30	974F														予	余	与
	4D40	975F	誉	輿	預	傭	幼	妖	容	庸	揚	揺	擁	曜	楊	様	洋	溶
	4D50	976F	熔	用	窯	羊	耀	葉	蓉	要	謡	踊	遙	陽	養	慾	抑	欲
	4D60	9780	沃	浴	翌	翼	淀											
ラ	4D60	9780						羅	螺	裸	来	莱	頼	雷	洛	絡	落	酪
	4D70	9790	乱	卵	嵐	欄	濫	藍	蘭	覧								
リ	4D70	9790									利	吏	履	李	梨	理	璃	
	4E20	979E		痢	裏	裡	里	離	陸	律	率	立	葎	掠	略	劉	流	溜
	4E30	97AE	琉	留	硫	粒	隆	竜	龍	侶	慮	旅	虜	了	亮	僚	両	凌
	4E40	97BE	寮	料	梁	涼	猟	療	瞭	稜	糧	良	諒	遼	量	陵	領	力
	4E50	97CE	緑	倫	厘	林	淋	燐	琳	臨	輪	隣	鱗	麟				
ル	4E50	97CE													瑠	塁	涙	累
	4E60	97DE	類															
レ	4E60	97DE		令	伶	例	冷	励	嶺	怜	玲	礼	苓	鈴	隷	零	霊	麗
	4E70	97EE	齢	暦	歴	列	劣	烈	裂	廉	恋	憐	漣	煉	簾	練	聯	
	4F20	983F		蓮	連	錬												
ロ	4F20	983F					呂	魯	櫓	炉	賂	路	露	労	婁	廊	弄	朗
	4F30	984F	楼	榔	浪	漏	牢	狼	籠	老	聾	蠟	郎	六	麓	禄	肋	録
	4F40	985F	論															
ワ	4F40	985F		倭	和	話	歪	賄	脇	惑	枠	鷲	亙	亘	鰐	詫	藁	蕨
	4F50	986F	椀	湾	碗	腕												
	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	JIS コード	シフト JIS コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
一	5020	989E		弌	丐	丕												
丨	5020	989E					个	丱										
丶	5020	989E							丶	丼								
丿	5020	989E									丿	乂	乖	乘				
乙	5020	989E													亂			
亅	5020	989E														亅	豫	亊
	5030	98AE	舒															
二	5030	98AE		弍	于	亞	亟											
亠	5030	98AE						亠	亢	亰	亳	亶						
人	5030	98AE											从	仍	仄	仆	仂	仗
	5040	98BE	仞	仭	仟	价	伉	佚	估	佛	佝	佗	佇	佶	侈	侏	侘	佻
	5050	98CE	佩	佰	侑	佯	來	侖	侭	俔	俟	俎	俘	俛	俑	俚	俐	俤
	5060	98DE	俥	倚	倨	倔	倪	倥	倅	伜	俶	倡	倩	倬	俾	俯	們	倆
	5070	98EE	偃	假	會	偕	偐	偈	做	偖	偬	偸	傀	傚	傅	傴	傲	
	5120	993F		僉	僊	傳	僂	僖	僞	僥	僭	僣	僮	價	僵	儉	儁	儂
	5130	994F	儖	儕	儔	儚	儡	儺	儷	儼	儻							
儿	5130	994F										儿	兀	兒	兌	兔	兢	競
入	5140	995F	兩	兪														
八	5140	995F			兮	冀												
冂	5140	995F					冂	囘	册	冉	冏	冑	冓	冕				
冖	5140	995F													冖	冤	冦	冢
	5150	996F	冩	冪														
	JIS コード	シフト JIS コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
冫	5150	996F			冫	决	冱	冲	冰	况	冽	凅	凉	凛				
几	5150	996F													几	處	凩	凭
	5160	9980	凰															
凵	5160	9980		凵	凾													
刀	5160	9980				刄	刋	刔	刎	刧	刪	刮	刳	刹	剏	剄	剋	剌
	5170	9990	剞	剔	剪	剴	剩	剳	剿	剽	劍	劔	劒	剱	劈	劑	辨	
	5220	999E		辧														
力	5220	999E			劬	劭	劼	劵	勁	勍	勗	勞	勣	勦	飭	勠	勳	勵
	5230	99AE	勸															
勹	5230	99AE		勹	匆	匈	甸	匍	匐	匏								
匕	5230	99AE									匕							
匚	5230	99AE										匚	匣	匯	匱	匳		
匸	5230	99AE															匸	區
十	5240	99BE	卆	卅	丗	卉	卍	凖										
卜	5240	99BE							卞									
卩	5240	99BE								卩	卮	夘	卻	卷				
厂	5240	99BE													厂	厖	厠	厦
	5250	99CE	厥	厮	厰													
厶	5250	99CE				厶	參	簒										
又	5250	99CE							雙	叟	曼	燮						
	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
口	5250	99CE											叮	叨	叭	叺	吁	吽
	5260	99DE	呀	听	吭	吼	吮	吶	吩	吝	呎	咏	呵	咎	呟	呱	呷	呰
	5270	99EE	咒	呻	咀	呶	咄	咐	咆	哇	咢	咸	咥	咬	哄	哈	咨	
	5320	9A3F		咫	哂	咤	咾	咼	哘	哥	哦	唏	唔	哽	哮	哭	哺	哢
	5330	9A4F	唹	啀	啣	啌	售	啜	啅	啖	啗	唸	唳	啝	喙	喀	咯	喊
	5340	9A5F	喟	啻	啾	喘	喞	單	啼	喃	喩	喇	喨	嗚	嗅	嗟	嗄	嗜
	5350	9A6F	嗤	嗔	嘔	嗷	嘖	嗾	嗽	嘛	嗹	噎	噐	營	嘴	嘶	嘲	嘸
	5360	9A80	噫	噤	嘯	噬	噪	嚆	嚀	嚊	嚠	嚔	嚏	嚥	嚮	嚶	嚴	囂
	5370	9A90	嚼	囁	囃	囀	囈	囎	囑	囓								
囗	5370	9A90									囗	囮	囹	圀	囿	圄	圉	
	5420	9A9E		圈	國	圍	圓	團	圖	嗇	圜							
土	5420	9A9E										圦	圷	圸	坎	圻	址	坏
	5430	9AAE	坩	埀	垈	坡	坿	垉	垓	垠	垳	垤	垪	垰	埃	埆	埔	埒
	5440	9ABE	埓	堊	埖	埣	堋	堙	堝	塲	堡	塢	塋	塰	毀	塒	堽	塹
	5450	9ACE	墅	墹	墟	墫	墺	壞	墻	墸	墮	壅	壓	壑	壗	壙	壘	壥
	5460	9ADE	壜	壤	壟													
士	5460	9ADE				壯	壺	壹	壻	壼	壽							
夂	5460	9ADE										夂						
夊	5460	9ADE											夊	夐				
夕	5460	9ADE													夛	梦	夥	
大	5460	9ADE																夬
	5470	9AEE	夭	夲	夸	夾	竒	奕	奐	奎	奚	奘	奢	奠	奧	奬	奩	
女	5520	9B3F		奸	妁	妝	佞	侫	妣	妲	姆	姨	姜	妍	姙	姚	娥	娟
	5530	9B4F	娑	娜	娉	娚	婀	婬	婉	娵	娶	婢	婪	媚	媼	媾	嫋	嫂
	5540	9B5F	媽	嫣	嫗	嫦	嫩	嫖	嫺	嫻	嬌	嬋	嬖	嬲	嫐	嬪	嬶	嬾
	5550	9B6F	孃	孅	孀													
	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	JIS コード	シフトJIS コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
子	5550	9B6F				孑	孕	孚	孛	孥	孩	孰	孳	孵	學	斈	孺	
宀	5550	9B6F																宀
	5560	9B80	它	宦	宸	寃	寇	寉	寔	寐	寤	實	寢	寞	寥	寫	寰	寶
	5570	9B90	寳															
寸	5570	9B90		尅	將	專	對											
小	5570	9B90						尓	尠									
尢	5570	9B90								尢	尨							
尸	5570	9B90										尸	尹	屁	屆	屎	屓	
	5620	9B9E		屐	屏	孱	屬											
屮	5620	9B9E						屮										
山	5620	9B9E							乢	屶	屹	岌	岑	岔	妛	岫	岻	岶
	5630	9BAE	岼	岷	峅	岾	峇	峙	峩	峽	峺	峭	嶌	峪	崋	崕	崗	嵜
	5640	9BBE	崟	崛	崑	崔	崢	崚	崙	崘	嵌	嵒	嵎	嵋	嵬	嵳	嵶	嶇
	5650	9BCE	嶄	嶂	嶢	嶝	嶬	嶮	嶽	嶐	嶷	嶼	巉	巍	巓	巒	巖	
巛	5650	9BCE																巛
工	5660	9BDE	巫															
己	5660	9BDE		已	巵													
巾	5660	9BDE				帋	帚	帙	帑	帛	帶	帷	幄	幃	幀	幎	幗	幔
	5670	9BEE	幟	幢	幤	幇												
干	5670	9BEE					幵	并										
幺	5670	9BEE							幺	麼								
	JIS コード	シフトJIS コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
广	5670	9BEE									广	庠	廁	廂	廈	廐	廏	
	5720	9C3F		廖	廣	廝	廚	廛	廢	廡	廨	廩	廬	廱	廳	廰		
廴	5720	9C3F															廴	廸
廾	5730	9C4F	廾	弃	弉	彝	彜											
弋	5730	9C4F						弋	弑									
弓	5730	9C4F								弖	弩	弭	弸	彁	彈	彌	彎	弯
彑	5740	9C5F	彑	彖	彗	彙												
彡	5740	9C5F					彡	彭										
彳	5740	9C5F							彳	彷	徃	徂	彿	徊	很	徑	徇	從
	5750	9C6F	徙	徘	徠	徨	徭	徼										
心	5750	9C6F							忖	忻	忤	忸	忱	忝	悳	忿	怡	恠
	5760	9C80	怙	怐	怩	怎	怱	怛	怕	怫	怦	怏	怺	恚	恁	恪	恷	恟
	5770	9C90	恊	恆	恍	恣	恃	恤	恂	恬	恫	恙	悁	悍	惧	悃	悚	
	5820	9C9E		悄	悛	悖	悗	悒	悧	悋	惡	悸	惠	惓	悴	忰	悽	惆
	5830	9CAE	悵	惘	慍	愕	愆	惶	惷	愀	惴	惺	愃	愡	惻	惱	愍	愎
	5840	9CBE	慇	愾	愨	愧	慊	愿	愼	愬	愴	愽	慂	慄	慳	慷	慘	慙
	5850	9CCE	慚	慫	慴	慯	慥	慱	慟	慝	慓	慵	憙	憖	憇	憬	憔	憚
	5860	9CDE	憊	憑	憫	憮	懌	懊	應	懷	懈	懃	懆	憺	懋	罹	懍	懦
	5870	9CEE	懣	懶	懺	懴	懿	懽	懼	懾	戀							
戈	5870	9CEE										戈	戉	戍	戌	戔	戛	
	5920	9D3F		戞	戡	截	戮	戰	戲	戳								
戸	5920	9D3F									扁							
手	5920	9D3F										扎	扞	扣	扛	扠	扨	扼
	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
手	5930	9D4F	扛	抉	找	抒	抓	抖	拔	抃	抔	拗	拑	抻	拏	拿	拆	擔
	5940	9D5F	拈	拜	拌	拊	拂	拇	抛	拉	挌	拮	拱	挧	挂	挈	拯	拵
	5950	9D6F	捐	挾	捍	搜	捏	掖	掎	掀	掫	捶	掣	掏	掉	掟	掵	捫
	5960	9D80	捩	掾	揩	揀	揆	揣	揉	插	揶	揄	搖	搴	搆	搓	搦	搶
	5970	9D90	攝	搗	搨	搏	摧	摯	摶	摎	攪	撕	撓	撥	撩	撈	撼	
	5A20	9D9E		據	擒	擅	擇	撻	擘	擂	擱	擧	舉	擠	擡	抬	擣	擯
	5A30	9DAE	攬	擶	擴	擲	擺	攀	擽	攘	攜	攅	攤	攣	攫			
支	5A30	9DAE														攴	攵	攷
	5A40	9DBE	收	攸	畋	效	敖	敕	敍	敘	敞	敝	敲	數	斂	斃	變	
斗	5A40	9DBE																斛
	5A50	9DCE	斟															
斤	5A50	9DCE		斫	斷													
方	5A50	9DCE				旃	旆	旁	旄	旌	旒	旛	旙					
无	5A50	9DCE												无	旡			
日	5A50	9DCE														旱	杲	昊
	5A60	9DDE	昃	旻	杳	昵	昶	昴	昜	晏	晄	晉	晁	晞	晝	晤	晧	晨
	5A70	9DEE	晟	晢	晰	暃	暈	暎	暉	暄	暘	暝	曁	暹	曉	暾	暼	
	5B20	9E3F		曄	暸	曖	曚	曠	昿	曦	曩							
曰	5B20	9E3F										曰	曵	曷				
月	5B20	9E3F													朏	朖	朞	朦
	5B30	9E4F	朧	霸														
木	5B30	9E4F			朮	朿	朶	杁	朸	朷	杆	杞	杠	杙	杣	杤	枉	杰
	5B40	9E5F	枩	杼	杪	枌	枋	枦	枡	枅	枷	柯	枴	柬	枳	柩	枸	柤
	5B50	9E6F	柞	柝	柢	柮	枹	柎	柆	柧	桧	栞	框	栩	桀	桍	栲	桎
	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
木	5B60	9E80	梳	栫	桙	档	桷	桿	梟	梏	梭	梔	條	梛	梃	檮	梹	桴
	5B70	9E90	梵	梠	梺	椏	梍	桾	椁	棊	椈	棘	椢	椦	棡	椌	棍	
	5C20	9E9E		棔	棧	棕	椶	椒	椄	棗	棣	椥	棹	棠	棯	椨	椪	椚
	5C30	9EAE	椣	椡	棆	楹	楷	楜	楸	楫	楔	楾	楮	椹	楴	椽	楙	椰
	5C40	9EBE	楡	楞	楝	榁	楪	榲	榮	槐	榿	槁	槓	榾	槎	寨	槊	槝
	5C50	9ECE	榻	槃	榧	樮	榑	榠	榜	榕	榴	槞	槨	樂	樛	槿	權	槹
	5C60	9EDE	槲	槧	樅	榱	樞	槭	樔	槫	樊	樒	櫁	樣	樓	橄	樌	橲
	5C70	9EEE	樶	橸	橇	橢	橙	橦	橈	樸	樢	檐	檍	檠	檄	檢	檣	
	5D20	9F3F		檗	蘗	檻	櫃	櫂	檸	檳	檬	櫞	櫑	櫟	檪	櫚	櫪	櫻
	5D30	9F4F	欅	蘖	櫺	欒	欖	鬱	欟									
欠	5D30	9F4F								欸	欷	盜	欹	飮	歇	歃	歉	歐
	5D40	9F5F	歙	歔	歛	歟	歡											
止	5D40	9F5F						歸										
歹	5D40	9F5F							歹	歿	殀	殄	殃	殍	殘	殕	殞	殤
	5D50	9F6F	殪	殫	殯	殲	殱											
殳	5D50	9F6F						殳	殷	殼	毆							
母	5D50	9F6F										毋	毓					
毛	5D50	9F6F												毟	毬	毫	毳	毯
	5D60	9F80	麾	氈														
氏	5D60	9F80			氓													
气	5D60	9F80				气	氛	氤	氣									
水	5D60	9F80								汞	汕	汢	汪	沂	沍	沚	沁	沛
	5D70	9F90	汾	汨	汳	沒	沐	泄	泱	泓	沽	泗	泅	泝	沮	沱	沾	
	5E20	9F9E		沺	泛	泯	泙	泪	洟	衍	洶	洫	洽	洸	洙	洵	洳	洒
	5E30	9FAE	洌	浣	涓	浤	浚	浹	浙	涎	涕	濤	涅	淹	渕	渊	涵	淇
	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
水	5E40	9FBE	淦	涸	淆	淬	淞	淌	淨	淒	淅	淺	淙	淤	淕	淪	淮	渭
	5E50	9FCE	湮	渮	渙	湲	湟	渾	渣	湫	渫	湶	湍	渟	湃	渺	湎	渤
	5E60	9FDE	滿	渝	游	溂	溪	溘	滉	溷	滓	溽	溯	滄	溲	滔	滕	溏
	5E70	9FEE	溥	滂	溟	潁	漑	灌	滬	滸	滾	漿	滲	漱	滯	漲	滌	
	5F20	E03F		漾	漓	滷	澆	潺	潸	澁	澀	潯	潛	濳	潭	澂	潼	潘
	5F30	E04F	澎	澑	濂	潦	澳	澣	澡	澤	澹	濆	澪	濟	濕	濬	濔	濘
	5F40	E05F	濱	濮	濛	瀉	瀋	濺	瀑	瀁	瀏	濾	瀛	瀚	潴	瀝	瀘	瀟
	5F50	E06F	瀰	瀾	瀲	灑	灣											
火	5F50	E06F						炙	炒	炯	烱	炬	炸	炳	炮	烟	烋	烝
	5F60	E080	烙	焉	烽	焜	焙	煥	煕	熈	煦	煢	煌	煖	煬	熏	燻	熄
	5F70	E090	熕	熨	熬	燗	熹	熾	燒	燉	燔	燎	燠	燬	燧	燵	燼	
	6020	E09E		燹	燿	爍	爐	爛	爨									
爪	6020	E09E								爭	爬	爰	爲					
爻	6020	E09E												爻	爼			
爿	6020	E09E														爿	牀	牆
	6030	E0AE	牋	牘														
牛	6030	E0AE			牴	牾	犂	犁	犇	犒	犖	犢	犧					
犬	6030	E0AE												犹	犲	狃	狆	狄
	6040	E0BE	狎	狒	狢	狠	狡	狹	狷	倏	猗	猊	猜	猖	猝	猴	猯	猩
	6050	E0CE	猥	猾	獎	獏	默	獗	獪	獨	獰	獸	獵	獻	獺			
玉	6050	E0CE														珈	玳	珎
	6060	E0DE	玻	珀	珥	珮	珞	璢	琅	瑯	琥	珸	琲	琺	瑕	琿	瑟	瑙
	6070	E0EE	瑁	瑜	瑩	瑰	瑣	瑪	瑶	瑾	璋	璞	璧	瓊	瓏	瓔	珱	
瓜	6120	E13F		瓠	瓣													
瓦	6120	E13F				瓧	瓩	瓮	瓲	瓰	瓱	瓸	瓷	甄	甃	甅	甌	甎
	6130	E14F	甍	甕	甓													
	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	JIS コード	シフトJIS コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
甘	6130	E14F				甞												
生	6130	E14F					甦											
用	6130	E14F						甬										
田	6130	E14F							甼	畄	畍	畊	畉	畛	畆	畚	畩	畤
	6140	E15F	畧	畫	畭	畸	當	疆	疇	畴	疊	疉	疂					
疒	6140	E15F												疔	疚	疝	疥	疣
	6150	E16F	痂	疳	痃	疵	疽	疸	疼	疱	痍	痊	痒	痙	痣	痞	痾	痿
	6160	E180	痼	瘁	痰	痺	痲	痳	瘋	瘍	瘉	瘟	瘧	瘠	瘡	瘢	瘤	瘴
	6170	E190	瘰	瘻	癇	癈	癆	癜	癘	癡	癢	癨	癩	癪	癧	癬	癰	
	6220	E19E		癲														
癶	6220	E19E			癶	癸	發											
白	6220	E19E						皀	皃	皈	皋	皎	皖	皓	皙	皚		
皮	6220	E19E															皰	皴
	6230	E1AE	皸	皹	皺													
皿	6230	E1AE				盂	盍	盖	盒	盞	盡	盥	盧	盪	蘯			
目	6230	E1AE														盻	眈	眇
	6240	E1BE	眄	眩	眤	眞	眥	眦	眛	眷	眸	睇	睚	睨	睫	睛	睥	睿
	6250	E1CE	睾	睹	瞎	瞋	瞑	瞠	瞞	瞰	瞶	瞹	瞿	瞼	瞽	瞻	矇	矍
	6260	E1DE	矗	矚														
矛	6260	E1DE			矜													
矢	6260	E1DE				矣	矮											
石	6260	E1DE						矼	砌	砒	礦	砠	礪	硅	碎	硴	碆	硼
	6270	E1EE	碚	碌	碣	碵	碪	碯	磑	磆	磋	磔	碾	碼	磅	磊	磬	
	JIS コード	シフトJIS コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
石	6320	E23F		磧	磚	磽	磴	礇	礒	礑	礙	礬	礫					
示	6320	E23F												祀	祠	祗	祟	祚
	6330	E24F	祕	祓	祺	祿	禊	禝	禧	齋	禪	禮	禳					
禸	6330	E24F												禹	禺			
禾	6330	E24F														秉	秕	秧
	6340	E25F	秬	秡	秣	稈	稍	稘	稙	稠	稟	禀	稱	稻	稾	稷	穃	穗
	6350	E26F	穉	穡	穢	穩	龝	穰										
穴	6350	E26F							穹	穽	窈	窗	窕	窘	窖	窩	竈	窰
	6360	E280	窶	竅	竄	窿	邃	竇	竊									
立	6360	E280								竍	竏	竕	竓	站	竚	竝	竡	竢
	6370	E290	竦	竭	竰													
竹	6370	E290				笂	笏	笊	笆	笳	笘	笙	笞	笵	笨	笶	筐	
	6420	E29E		筺	笄	筍	笋	筌	筅	筵	筥	筴	筧	筰	筱	筬	筮	箝
	6430	E2AE	箘	箟	箍	箜	箚	箋	箒	箏	筝	箙	篋	篁	篌	篏	箴	篆
	6440	E2BE	篝	篩	簑	簔	篦	篥	篭	簀	簇	簓	篳	篷	簗	簍	篶	簣
	6450	E2CE	簧	簪	簟	簷	簫	簽	籌	籃	籔	籏	籀	籐	籘	籟	籤	籖
	6460	E2DE	籥	籬														
米	6460	E2DE			籵	粃	粐	粤	粭	粢	粫	粡	粨	粳	粲	粱	粮	粹
	6470	E2EE	粽	糀	糅	糂	糘	糒	糜	糢	鬻	糯	糲	糴	糶			
糸	6470	E2EE														糺	紆	
	6520	E33F		紂	紜	紕	紊	絅	絋	紮	紲	紿	紵	絆	絳	絖	絎	絲
	6530	E34F	絨	絮	絏	絣	經	綉	絛	綏	絽	綛	綺	綮	綣	綵	緇	綽
	6540	E35F	綫	總	綢	綯	緜	綸	綟	綰	緘	緝	緤	緞	緻	緲	緡	縅
	6550	E36F	縊	縣	縡	縒	縱	縟	縉	縋	縢	繆	繦	縻	縵	縹	繃	縷
	6560	E380	縲	縺	繧	繝	繖	繞	繙	繚	繹	繪	繩	繼	繻	纃	緕	繽
	6570	E390	辮	繿	纈	纉	續	纒	纐	纓	纔	纖	纎	纛	纜			
	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
缶	6570	E390														缸	缺	
	6620	E39E		罅	罌	罍	罎	罐										
网	6620	E39E							网	罕	罔	罘	罟	罠	罨	罩	罧	罸
	6630	E3AE	羂	羆	羃	羈	羇											
羊	6630	E3AE						羌	羔	羞	羝	羚	羣	羯	羲	羹	羮	羶
	6640	E3BE	羸	譱														
羽	6640	E3BE			翅	翆	翊	翕	翔	翡	翦	翩	翳	翹	飜			
老	6640	E3BE														耆	耄	耋
耒	6650	E3CE	耒	耘	耙	耜	耡	耨										
耳	6650	E3CE							耿	耻	聊	聆	聒	聘	聚	聟	聢	聨
	6660	E3DE	聳	聲	聰	聶	聹	聽										
聿	6660	E3DE							聿	肄	肆	肅						
肉	6660	E3DE											肛	肓	肚	肭	冐	肬
	6670	E3EE	胛	胥	胙	胝	胄	胚	胖	脉	胯	胱	脛	脩	脣	脯	腋	
	6720	E43F		隋	腆	脾	腓	腑	胼	腱	腮	腥	腦	腴	膃	膈	膊	膀
	6730	E44F	膂	膠	膕	膤	膣	腟	膓	膩	膰	膵	膾	膸	膽	臀	臂	膺
	6740	E45F	臉	臍	臑	臙	臘	臈	臚	臟	臠							
臣	6740	E45F										臧						
至	6740	E45F											臺	臻				
臼	6740	E45F													臾	舁	舂	舅
	6750	E46F	與	舊														
舌	6750	E46F			舍	舐	舖											
	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
舟	6750	E46F						舩	舫	舸	舳	艀	艙	艘	艝	艚	艟	艤
	6760	E480	艢	艨	艪	艫	舮											
艮	6760	E480						艱										
色	6760	E480							艷									
艸	6760	E480								艸	艾	芍	芒	芫	芟	芻	芬	苡
	6770	E490	苣	苟	苒	苴	苳	苺	莓	范	苻	苹	苞	茆	苜	茉	苙	
	6820	E49E		茵	茴	茖	茲	茱	荀	茹	荐	荅	茯	茫	茗	茘	莅	莚
	6830	E4AE	莪	莟	莢	莖	茣	莎	莇	莊	荼	莵	荳	荵	莠	莉	莨	菴
	6840	E4BE	萓	菫	菎	菽	萃	菘	萋	菁	菷	萇	菠	菲	萍	萢	萠	莽
	6850	E4CE	萸	蔆	菻	葭	萪	萼	蕚	蒄	葷	葫	蒭	葮	蒂	葩	葆	萬
	6860	E4DE	葯	葹	萵	蓊	葢	蒹	蒿	蒟	蓙	蓍	蒻	蓚	蓐	蓁	蓆	蓖
	6870	E4EE	蒡	蔡	蓿	蓴	蔗	蔘	蔬	蔟	蔕	蔔	蓼	蕀	蕣	蕘	蕈	
	6920	E53F		蕁	蘂	蕋	蕕	薀	薤	薈	薑	薊	薨	蕭	薔	薛	藪	薇
	6930	E54F	薜	蕷	蕾	薐	藉	薺	藏	薹	藐	藕	藝	藥	藜	藹	蘊	蘓
	6940	E55F	蘋	藾	藺	蘆	蘢	蘚	蘰	蘿								
虍	6940	E55F									虍	乕	虔	號	虧			
虫	6940	E55F														虱	蚓	蚣
	6950	E56F	蚩	蚪	蚋	蚌	蚶	蚯	蛄	蛆	蚰	蛉	蠣	蚫	蛔	蛞	蛩	蛬
	6960	E580	蛟	蛛	蛯	蜒	蜆	蜈	蜀	蜃	蛻	蜑	蜉	蜍	蛹	蜊	蜴	蜿
	6970	E590	蜷	蜻	蜥	蜩	蜚	蝠	蝟	蝸	蝌	蝎	蝴	蝗	蝨	蝮	蝙	
	6A20	E59E		蝓	蝣	蝪	蠅	螢	螟	螂	螯	蟋	螽	蟀	蟐	雖	螫	蟄
	6A30	E5AE	螳	蟇	蟆	螻	蟯	蟲	蟠	蠏	蠍	蟾	蟶	蟷	蠎	蟒	蠑	蠖
	6A40	E5BE	蠕	蠢	蠡	蠱	蠶	蠹	蠧	蠻								
血	6A40	E5BE									衄	衂						
行	6A40	E5BE											衒	衙	衞	衢		
衣	6A40	E5BE															衫	袁
	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
衣	6A50	E5CE	衾	衮	衵	衽	袵	衲	袂	袗	袒	袮	袙	袢	袍	袤	袰	袿
	6A60	E5DE	袱	裃	裄	裔	裘	裙	裝	裹	褂	裼	裴	裨	裲	褄	褌	褊
	6A70	E5EE	褓	襃	褞	褥	褪	褫	襁	襄	褻	褶	褸	襌	褝	襠	襞	
	6B20	E63F		襦	襤	襭	襪	襯	襴	襷								
襾	6B20	E63F									襾	覃	覈	覊				
見	6B20	E63F													覓	覘	覡	覩
	6B30	E64F	覦	覬	覯	覲	覺	覽	覿	觀								
角	6B30	E64F									觚	觜	觝	觧	觴	觸		
言	6B30	E64F															訃	訖
	6B40	E65F	訐	訌	訛	訝	訥	訶	詁	詛	詒	詆	詈	詼	詭	詬	詢	誅
	6B50	E66F	誂	誄	誨	誡	誑	誥	誦	誚	誣	諄	諍	諂	諚	諫	諳	諧
	6B60	E680	諤	諱	謔	諠	諢	諷	諞	諛	謌	謇	謚	諡	謖	謐	謗	謠
	6B70	E690	謳	鞫	謦	謫	謾	謨	譁	譌	譏	譎	證	譖	譛	譚	譫	
	6C20	E69E		譟	譬	譯	譴	譽	讀	讌	讎	讒	讓	讖	讙	讚		
谷	6C20	E69E															谺	豁
	6C30	E6AE	谿															
豆	6C30	E6AE		豈	豌	豎	豐											
豕	6C30	E6AE						豕	豢	豬								
豸	6C30	E6AE									豸	豺	貂	貉	貅	貊	貍	貎
	6C40	E6BE	貔	豼	貘													
貝	6C40	E6BE				戝	貭	貪	貽	貲	貳	貮	貶	賈	賁	賤	賣	賚
	6C50	E6CE	賽	賺	賻	贄	贅	贊	贇	贏	贍	贐	齎	贓	賍	贔	贖	
赤	6C50	E6CE																赧
	6C60	E6DE	赭															
	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

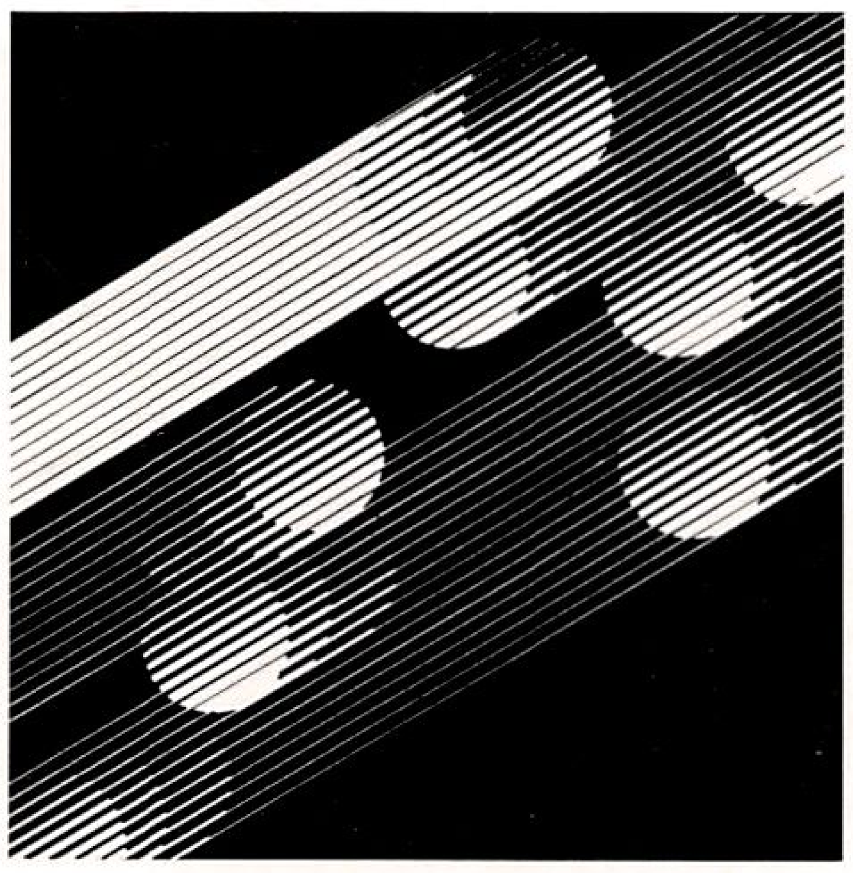
	JIS コード	シフトJIS コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
走	6C60	E6DE		赱	赳	趁	趙											
足	6C60	E6DE						跂	趾	趺	跏	跚	跖	跌	跛	跋	跪	跫
	6C70	E6EE	跟	跣	跼	踈	踉	跿	踝	踞	踐	踟	蹂	踵	踰	踴	蹊	
	6D20	E73F		蹇	蹉	蹌	蹐	蹈	蹙	蹤	蹠	踪	蹣	蹕	蹶	蹲	蹼	躁
	6D30	E74F	躇	躅	躄	躋	躊	躓	躑	躔	躙	躪	躡					
身	6D30	E74F												躬	躰	軆	躱	躾
	6D40	E75F	軅	軈														
車	6D40	E75F			軋	軛	軣	軼	軻	軫	軾	輊	輅	輕	輒	輙	輓	輜
	6D50	E76F	輟	輛	輌	輦	輳	輻	輹	轅	轂	輾	轌	轉	轆	轎	轗	轜
	6D60	E780	轢	轣	轤													
辛	6D60	E780				辜	辟	辣	辭	辯								
辶	6D60	E780									辷	迚	迥	迢	迪	迯	迩	迴
	6D70	E790	逅	迹	迺	逑	逕	逡	逍	逞	逖	逋	逧	逶	逵	逹	迸	
	6E20	E79E		遏	遐	遑	遒	逎	遉	逾	遖	遘	遞	遨	遯	遶	隨	遲
	6E30	E7AE	邂	遽	邁	邀	邊	邉	邏									
邑	6E30	E7AE								邨	邯	邱	邵	郢	郤	扈	郛	鄂
	6E40	E7BE	鄒	鄙	鄲	鄰												
酉	6E40	E7BE					酊	酖	酘	酣	酥	酩	酳	酲	醋	醉	醂	醢
	6E50	E7CE	醫	醯	醪	醵	醴	醺	釀	釁								
釆	6E50	E7CE									釉	釋						
里	6E50	E7CE											釐					
金	6E50	E7CE												釖	釟	釡	釛	釼
	6E60	E7DE	釵	釶	鈞	釿	鈔	鈬	鈕	鈑	鉞	鉗	鉅	鉉	鉤	鉈	銕	鈿
	JIS コード	シフトJIS コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
金	6E70	E7EE	鉋	鉐	銜	銖	銓	銛	鉚	鋏	銹	銷	鋩	錏	鋺	鍄	錮	
	6F20	E83F		錙	錢	錚	錣	錺	錵	錻	鍜	鍠	鍼	鍮	鍖	鎰	鎬	鎭
	6F30	E84F	鎔	鎹	鏖	鏗	鏨	鏥	鏘	鏃	鏝	鏐	鏈	鏤	鐚	鐔	鐓	鐃
	6F40	E85F	鐇	鐐	鐶	鐫	鐵	鐡	鐺	鑁	鑒	鑄	鑛	鑠	鑢	鑞	鑪	鈩
	6F50	E86F	鑰	鑵	鑷	鑽	鑚	鑼	鑾	钁	鑿							
門	6F50	E86F										閂	閇	閊	閔	閖	閘	閙
	6F60	E880	閠	閨	閧	閭	閼	閻	閹	閾	闊	濶	闃	闍	闌	闕	闔	闖
	6F70	E890	關	闡	闥	闢												
阜	6F70	E890					阡	阨	阮	阯	陂	陌	陏	陋	陷	陜	陞	
	7020	E89E		陝	陟	陦	陲	陬	隍	隘	隕	隗	險	隧	隱	隲	隰	隴
隶	7030	E8AE	隶	隸														
隹	7030	E8AE			隹	雎	雋	雉	雍	襍	雜	霍	雕					
雨	7030	E8AE												雹	霄	霆	霈	霓
	7040	E8BE	霎	霑	霏	霖	霙	霤	霪	霰	霹	霽	霾	靄	靆	靈	靂	靉
青	7050	E8CE	靜															
非	7050	E8CE		靠														
面	7050	E8CE			靤	靦	靨											
革	7050	E8CE						勒	靫	靱	靹	鞅	靼	鞁	靺	鞆	鞋	鞏
	7060	E8DE	鞐	鞜	鞨	鞦	鞣	鞳	鞴	韃	韆	韈						
韋	7060	E8DE											韋	韜				
韭	7060	E8DE													韭	齏	韲	
音	7060	E8DE																竟
	7070	E8EE	韶	韵														
	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
頁	7070	E8EE			頏	頌	頸	頤	頡	頷	頽	顆	顏	顋	顫	顯	顰	
	7120	E93F		顱	顴	顳												
風	7120	E93F					颪	颯	颱	颶	飄	飃	飆					
食	7120	E93F												飩	飫	餃	餉	餒
	7130	E94F	餔	餘	餡	餝	餞	餤	餠	餬	餮	餽	餾	饂	饉	饅	饐	饋
	7140	E95F	饑	饒	饌	饕												
首	7140	E95F					馗	馘										
香	7140	E95F							馥									
馬	7140	E95F								馭	馮	馼	駟	駛	駝	駘	駑	駭
	7150	E96F	駮	駱	駲	駻	駸	騁	騏	騅	駢	騙	騫	騷	驅	驂	驀	驃
	7160	E980	騾	驕	驍	驛	驗	驟	驢	驥	驤	驩	驫	驪				
骨	7160	E980													骭	骰	骼	髀
	7170	E990	髏	髑	髓	體												
高	7170	E990					髞											
髟	7170	E990						髟	髢	髣	髦	髯	髫	髮	髴	髱	髷	
	7220	E99E		髻	鬆	鬘	鬚	鬟	鬢	鬣								
鬥	7220	E99E									鬥	鬧	鬨	鬩	鬪	鬮		
鬯	7220	E99E															鬯	
鬲	7220	E99E																鬲
鬼	7230	E9AE	魄	魃	魏	魍	魎	魑	魘									
魚	7230	E9AE								魴	鮓	鮃	鮑	鮖	鮗	鮟	鮠	鮨
	7240	E9BE	鮴	鯀	鯊	鮹	鯆	鯏	鯑	鯒	鯣	鯢	鯤	鯔	鯡	鰺	鯲	鯱
	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
魚	7250	E9CE	鯰	鰕	鰔	鰉	鰓	鰌	鰆	鰈	鰒	鰊	鰄	鰮	鰛	鰥	鰤	鰡
	7260	E9DE	鰰	鱇	鰲	鱆	鰾	鱚	鱠	鱧	鱶	鱸						
鳥	7260	E9DE											鳧	鳬	鳰	鴉	鴈	鳫
	7270	E9EE	鴃	鴆	鴪	鴦	鶯	鴣	鴟	鵄	鴕	鴒	鵁	鴿	鴾	鵆	鵈	
	7320	EA3F		鵝	鵞	鵤	鵑	鵐	鵙	鵲	鶉	鶇	鶫	鵯	鵺	鶚	鶤	鶩
	7330	EA4F	鶲	鷄	鷁	鶻	鶸	鶺	鷆	鷏	鷂	鷙	鷓	鷸	鷦	鷭	鷯	鷽
	7340	EA5F	鸚	鸛	鸞													
鹵	7340	EA5F				鹵	鹹	鹽										
鹿	7340	EA5F							麁	麈	麋	麌	麒	麕	麑	麝		
麦	7340	EA5F															麥	麩
	7350	EA6F	麸	麪	麭													
麻	7350	EA6F				靡												
黄	7350	EA6F					黌											
黍	7350	EA6F						黎	黏	黐								
黒	7350	EA6F									黔	黜	點	黝	黠	黥	黨	黯
	7360	EA80	黴	黶	黷													
黹	7360	EA80				黹	黻	黼										
黽	7360	EA80							黽	鼇	鼈							
皷	7360	EA80										皷	鼕					
鼠	7360	EA80												鼡	鼬			
鼻	7360	EA80														鼾		
齊	7360	EA80															齊	
	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
齒	7360	EA80																齒
	7370	EA90	齔	齣	齟	齠	齡	齦	齧	齬	齪	齷	齲	齶				
龍	7370	EA90													龕			
龜	7370	EA90														龜		
龠	7370	EA90															龠	
	JISコード	シフトJISコード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F



NEC 日本電気株式会社・日本電気ホームエレクトロニクス株式会社

78118601